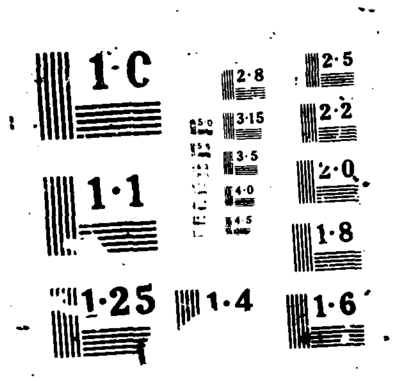
AD-A194 734 PRELIMINARY DESIGN AND CYCLE VERIFICATION OF A DIGITAL AUTOPILOT FOR AUTONOMOUS UNDERWATER VEHICLES(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S W DELAPLANE MAR 88 1/1

UNCLASSIFIED F/G 13/10.1 NL

END
DATE FILMED
8 88
DTIC


1·0
2·8
2·5
3·15
2·2
3·5
2·0
4·0
1·1
4·5
1·8
1·25
1·4
1·6

(2)

AD-A194 734

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC FILE COPY

THESIS

PRELIMINARY DESIGN AND CYCLE VERIFICATION OF A DIGITAL AUTOPILOT FOR AUTONOMOUS UNDERWATER VEHICLES

by

Stephen W. Delaplane

March 1988

Thesis Advisor: A.J. Healey

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AI

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
2b DECLASSIFICATION DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) Code 69	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
PRELIMINARY DESICN AND CYCLE VERIFICATION OF A DIGITAL AUTOPILOT FOR AUTONOMOUS UNDERWATER VEHICLES

12 PERSONAL AUTHOR(S)
Delaplane, Stephen W.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1988, March	69

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Digital Autopilot, for Autonomous Underwater Vehicles; Design

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Autonomous Underwater Vehicles (AUV's) are being considered for a diversity of U.S. Navy missions. They protend the advantages of cost effective fleet multiplication, minimal detectability, and reduced risk to personnel and high-value fleet assets. In response to the Department of the Navy and DARPA interests, AUV research and development by a number of public and private sector organizations has intensified in recent years. The Naval Postgraduate School has now built the first of a series of AUV models which will be used as "test-beds" for evolving automated control technologies. This thesis documents the results of initial efforts to install an on-site facility to support the design and development of a model-based digital autopilot for control of the NPS test vehicle. Using an IBM PC/AT and analog-digital interfacing, the development methodology has been implemented and verified by a simplified model reference control program for AUV dive plane commands.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassif
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. A.J. Healey	22b TELEPHONE (Include Area Code) (408) 646-2586; 22c OFFICE SYMBOL Code 69Hy

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office 1986—606-24

UNCLASSIFIED

Preliminary Design and Cycle Verification of a Digital Autopilot for Autonomous Underwater Vehicles

by

Stephen W. Delaplane
Commander, United States Navy
B.S., Purdue University, 1969
B.S., Old Dominion University, 1978

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1988

Author: [illegible]
Stephen W. Delaplane

Approved by: [illegible]
A.J. Healey, Thesis Advisor

[illegible]
A.J. Healey, Chairman
Department of Mechanical Engineering

[illegible]
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

Autonomous Underwater Vehicles (AUV's) are being considered for a diversity of U.S. Navy missions. They protend the advantages of cost effective fleet multiplication, minimal detectability, and reduced risk to personnel and high-value fleet assets. In response to the Department of the Navy and DARPA interests, AUV research and development by a number of public and private sector organizations has intensified in recent years. The Naval Postgraduate School has now built the first of a series of AUV models which will be used as "test-beds" for evolving automated control technologies. This thesis documents the results of initial efforts to install an on-site facility to support the design and development of a model-based digital autopilot for control of the NPS test vehicle. Using an IBM PC/AT and analog-digital interfacing, the development methodology has been implemented and verified by a simplified model reference control program for AUV dive plane commands.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

The concept of a hierarchical automated intelligent controller, or autopilot, as a state-of-the-art design methodology is clearly evident from the proceedings of [Ref. 1]. As a part of an inter-disciplinary research program, the Naval Postgraduate School is investigating many facets of automated control technologies for underwater vehicles. One of the objectives of this program has been to develop operating "test-bed" vehicles and the facilities for developing and evaluating real-time control systems.

In support of this objective, recent studies have led to the development of a model-referenced control methodology. Using the equations of motion for a vehicle of known hydrodynamic characteristics, computer simulation was used to develop the command generation logic, the design of a model following autopilot and evaluate the performance of the selected AUV computer model [Ref. 2].

Based on this vehicle's characteristics, subsequent research led to the design and construction of the first of a series of operational AUV models which will be used to evaluate evolving automated control technologies [Ref. 3]. This model was used to develop a technique for the identification of discrete transfer function relationships.

In-water tank tests were conducted for maneuvers in the vertical plane and from these, data was acquired and compared with the computer model simulation. A system transfer function relating dive plane commands to vehicle response was then determined.

B. RESEARCH OBJECTIVES

The focus of this research has been to implement a microcomputer based capability to support AUV data acquisition, vehicle response and transfer function identification, and the development of real time digital control programming.

Specific objectives included:

1. The acquisition and integration of available hardware and software.
2. The documentation of specific implementations and configurations used for the various phases of program development to date.
3. The development of a high-level language program for real-time control of the AUV test vehicle dive plane maneuvering based on the methodology and results of the research reported [Refs. 2,3]. This program would affirm the on-site development procedure and identify insufficiencies or incompatibilities which would need to be resolved to support future research.

This thesis documents the accomplishment of these objectives. The integration of hardware and software are discussed and evaluate in the overall perspective of the program development process. A program was developed and verified. This control program was written in Turbo Pascal and processes three analog sensor signals and issues a

digital diveplane command. The development procedure has been verified, however certain shortcomings in multi-channel signal processing and software compatability are discussed.

II. EQUIPMENT DESCRIPTION, APPLICATION AND PROCEDURAL DOCUMENTATION

A. THE DESIGN AND CONTROL PROGRAM DEVELOPMENT PROCESS: AN OVERVIEW

The development of real-time autonomous control programs is a dynamic and evolutionary process. It is the link between computer simulations of the systems and the implementation of controllers to achieve desired performance of these entities in the real-world. Autonomous controllers are exceedingly complex. In the early stages of controller development, software implementations operating in micro-computers which interface the device afford an opportunity for verification and optimization of design methodology and parameters. Once designs have been evaluated, the design logic is then implemented in an onboard controller. In this final implementation, much of the control program coding will have been replaced by functionally equivalent micro-electronic devices which permit economies in size and power as well as flexibility of application. The remaining control logic is micro-coded and programmed on micro electronic storage devices.

A principal objective of this research was the integration of available software and hardware to support the various activities in control program development for the AUV. These activities fall into three broad categories:

Data Acquisition, Data Analysis, and Design and Implementation of real-time control programs.

- Data Acquisition involves the assimilation of vehicle parameters and responses to known inputs under controlled conditions.

- Data Analysis involves the analysis of vehicle performance data for the purpose of extrapolating quantitative relationships between input commands and vehicle responses. These relationships are then reflected as gain factors in transfer functions which describe the AUV behavior or state.

- Design and Implementation of real-time control programs entails the representation of controller logic in programming code. This software operates in a micro-computer suitably interfaced with the AUV to permit the sampling of vehicle sensor data and to issue appropriate commands.

This chapter summarizes the specifications of the hardware and the software utilized to support these phases in the research reported herein. The phases and application of these resources are depicted in Figure 2.1. Concluding sections of this chapter document specific configurations and commentary which supplement manufacturer's technical reference materials with regard to this research application. Comments regarding the performance and suitability of these resources are reported in Chapter IV.

B. MICROCOMPUTER DESCRIPTION

An IBM PC/AT (compatible) microcomputer was acquired for this and future research. This computer operates at 6 or 10 MHz and with 0 or 1 wait states. An Intel 80286 microprocessor and an 80287 math co-processor were installed. The Random Access Memory (RAM) is expandable

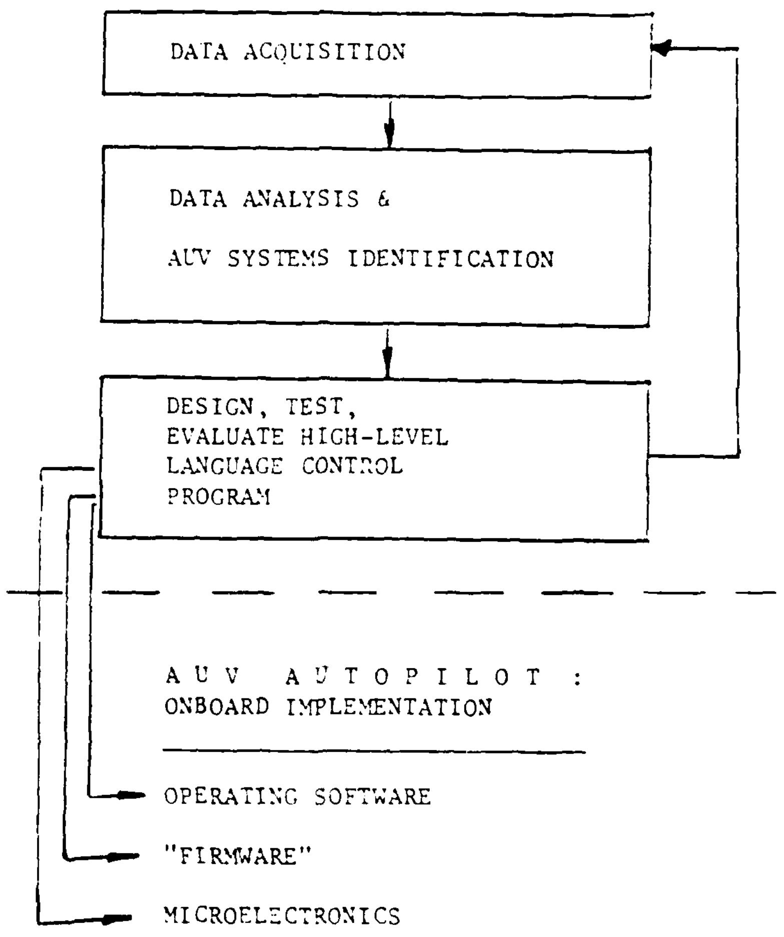

DATA ACQUISITION
DATA ANALYSIS &
AUV SYSTEMS IDENTIFICATION
DESIGN, TEST, EVALUATE HIGH-LEVEL LANGUAGE CONTROL PROGRAM
A U V A U T O P I L O T :
ONBOARD IMPLEMENTATION
OPERATING SOFTWARE
"FIRMWARE"
MICROELECTRONICS

Figure 2.1 Phases of Program Development

from its current size of 640 Kbytes. Installed floppy disk drives include a 1.2M byte (high density) and 360K byte floppy disk drives are installed. The floppy disk controller has the capability of controlling an additional 3.5 inch disk drive as well. A 40M byte hard disk is also installed.

C. SIGNAL PROCESSING: ANALOG-DIGITAL INTERFACE

Interfacing the micro-computer and the AUV is critical to the program development process. At this interface analog AUV sensor signals are digitized for the computer control program through an A/D conversion process. Similarly, digitized command signals are converted to analog equivalent signals by D/A conversion processing prior to transmission to the AUV analog servo-control components. The A/D signal processing was utilized exclusively during AUV test trials for data acquisition. Both modes of signal processing were utilized in development of the control program. A detailed description of the configurations during these phases are reported later in this chapter. This section concludes with a description of the devices used to achieve the analog-digital signal processing interface and a summary of the technical specifications detailed in the manufacturer's literature [Ref. 4].

Interfacing the AUV and the computer was accomplished using two modern device boards. The model DT 2801-A is a programmable,single board, analog and digital I/O signal processor. This board is one of a series designed for the IBM Personal Computers and was installed into one of the backplane slots on the computer's mother board. Once installed, a multi-pin connector, accessible at the back of the computer, facilitates connecting the DT 2801-A to an external screw terminal panel, the DT 707, by a ribbon cable and connector. The DT 707 was used to connect AUV sensor inputs, the command output, and an external trigger signal. Specific configurations are discussed later in this chapter.

The following summarizes detailed technical specifications contained in the manufacturer's product literature [Ref. 4].

1. DT 2801-A Digital and Analog Signal Processor

This device supports A/D conversion processing on 8 differential analog or 16 single ended, unipolar or bipolar, input channels with 12-bit resolution. Similarly, there are 2 D/A conversion channels for processing output signals. Software selectable gains accommodate a range of input signals. With 12-bit resolution and programmable gains it is possible to achieve a conversion accuracy on the order of +/- 0.03% of Full Scale Reading (FSR). An onboard clock can be used to initiate data conversion events for periods of 2.5 microseconds to 0.819 seconds in 1.25 microsecond increments. For more precise initiation of data conversion events or to synchronize with control program timing, an external trigger input is also available. Throughput, the time to execute one complete read or write cycle, of 27.5 KHz can be achieved under Direct Memory Access (DMA) procedures.

The onboard processor is programmable by any language that can access I/O registers of the IBM PC. Also, pre-programmed, user callable, subroutines are available for use with this device under the product name PCLAB. [Ref. 5] Similarly, a complete Data Acquisition software package, DT/NOTEBOOK is also available. [Ref. 6] These products were used in this research and are described in later sections of this chapter and in chapter 3.

2. DT 707 Screw Terminal Panel

This unit along with the ribbon cable and connector facilitate connection of I/O channels for analog to digital signal process and digital to digital signal processing. There are also terminal connections for external trigger signal sources.

D. SOFTWARE INTERFACING, ANALYSIS AND PROGRAMMING

1. Data Acquisition

This phase of program development was greatly facilitated by the use of a software program, DT/NOTEBOOK. [Ref. 6] The details of its employment and the system configuration are reported in the next section.

2. Data Analysis and AUV System Identification

Assuring the capability to analyze vehicle test data and identify discrete transfer function relationships was a principal objective of this research. This analysis was accomplished and reported in [Ref. 3] using a CAD Program called MATRIXx [Ref. 7] This program affords an extensive set of design and analysis functions for classical and state-space control. It also provides a state-of-the-art matrix analysis and graphical display capability.

3. Control Programming Utilities and Language

PC LAB consists of a library of low level routines which are callable as procedures, functions or subroutines in the following high level programming languages: BASIC, FORTRAN, Pascal, Turbo Pascal and C. These routines include single and multiple channel, or block, A/D and D/A conversion, with buffered storage and DMA control options. [Ref. 5] It should be noted that multiplexing of selected inputs using multiple channel, or block read, procedures is on the order of microseconds when using the DT 2801-A signal processing board. [Ref. 4] This sample interval was considered to adequately simulate "instantaneous" sampling of discrete vehicle parameters. Another signal processor, the DT 2818, is advertised to have "simultaneous sample and hold" and multiplexes the input channels on the order of nanoseconds. This processor may have application in future program development.

The control program reported in Chapter III was written in Turbo Pascal and compiled to run under either PC DOS or MS DOS operation systems. [Ref. 8] The facility of the Turbo Pascal program development environment, the intrinsic ability to manipulate text and data, the screen graphics capabilities and recursive procedure calls were strong considerations in the selection of this programming language. Discussion regarding use of specific PC LAB routines and Turbo Pascal programming are presented in Chapter III.

E. DOCUMENTATION AND CONFIGURATION FOR AUV DATA ACQUISITION

This section documents the configuration of hardware and provides commentary on the application of the DT/NOTEBOOK program as it was used in acquiring data during actual tank testing. Figure 2.2 depicts the system configuration for this phase of the research.

The DT/NOTEBOOK is a menu-driven data acquisition program with options for data analysis. (These options were not used, however, in favor of the more powerful CAD program MATRIXx previously discussed.) The DT/NOTEBOOK program affords a high level user interface with the DT 2801-A signal processor and was used exclusively for AUV data acquisition during tank testing. The comments in this section are intended to supplement the excellent tutorial and user documentation provided with the DT/NOTEBOOK program [Ref. 6] with comments specific to this research.

During the AUV trials three channels of analog data were sampled, digitized and stored for subsequent analysis. The three channels of sensor data consisted of depth, from a

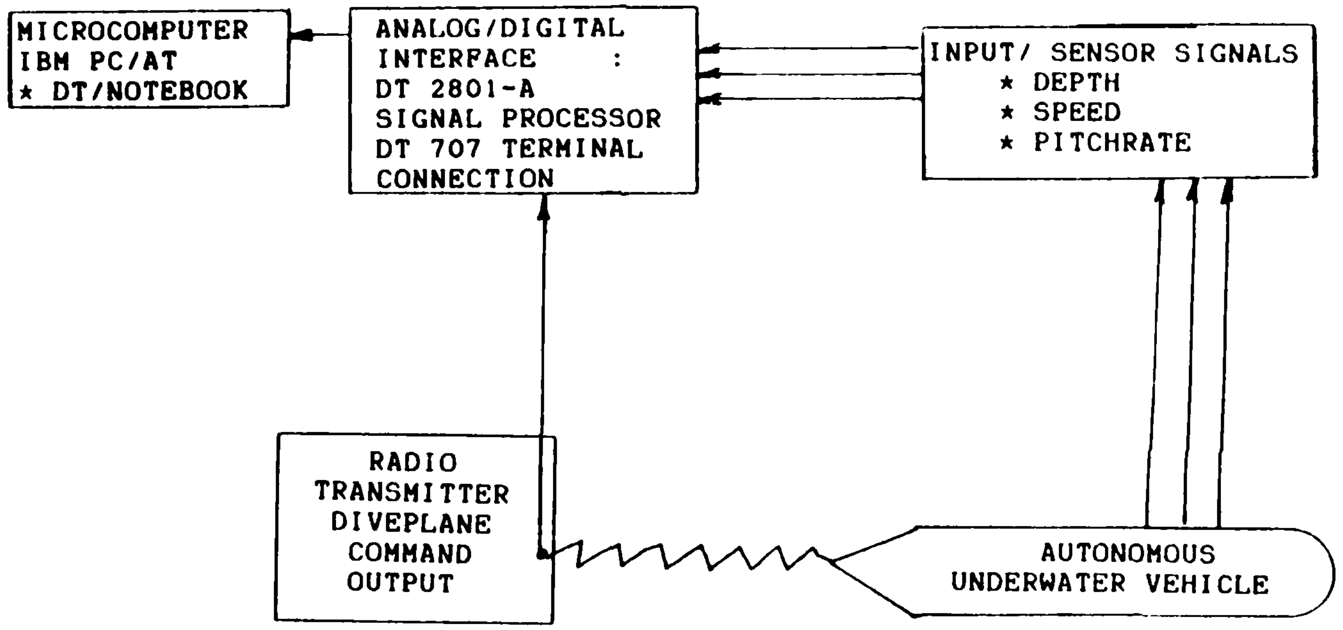


Figure 2.2 System Configuration for Data Acquisition

transducer output; speed from a dynamic head transducer, and pitchrate from a pitchrate gyro. The configuration of the DT 2801-A/DT 707 terminal panel interface with the AUV is shown in Figure 2.3.

DT/NOTEBOOK uses a hierarchical menu concept much like Lotus 123. Menu selections are made by either moving the highlighted input box with the cursor keys until it "captures" the desired option and then pressing the <Enter> or by merely pressing the key corresponding to the first letter of the command name. As successive options are made, the user moves deeper into the menu hierarchy. At each level, a new menu of commands or data input options is presented. Pressing the <EXC> key moves the user back out of the menu one level at a time.

Figure 2.4 illustrates the Main Menu. The program is contained on 12 disks. The first of these is a "Key Disk" which must be inserted in Drive A during installation and whenever the program is first executed. Installation instructions are clear and specific and are selected from the Main Menu using the INSTALL command. During the installation, two directories are created on the hard disk: Drivers and Notebook. Once the installation has been completed, the user merely locates himself in the Notebook directory, inserts the Master Key in Drive A, and enters "nb" from the keyboard. The program then displays the Main

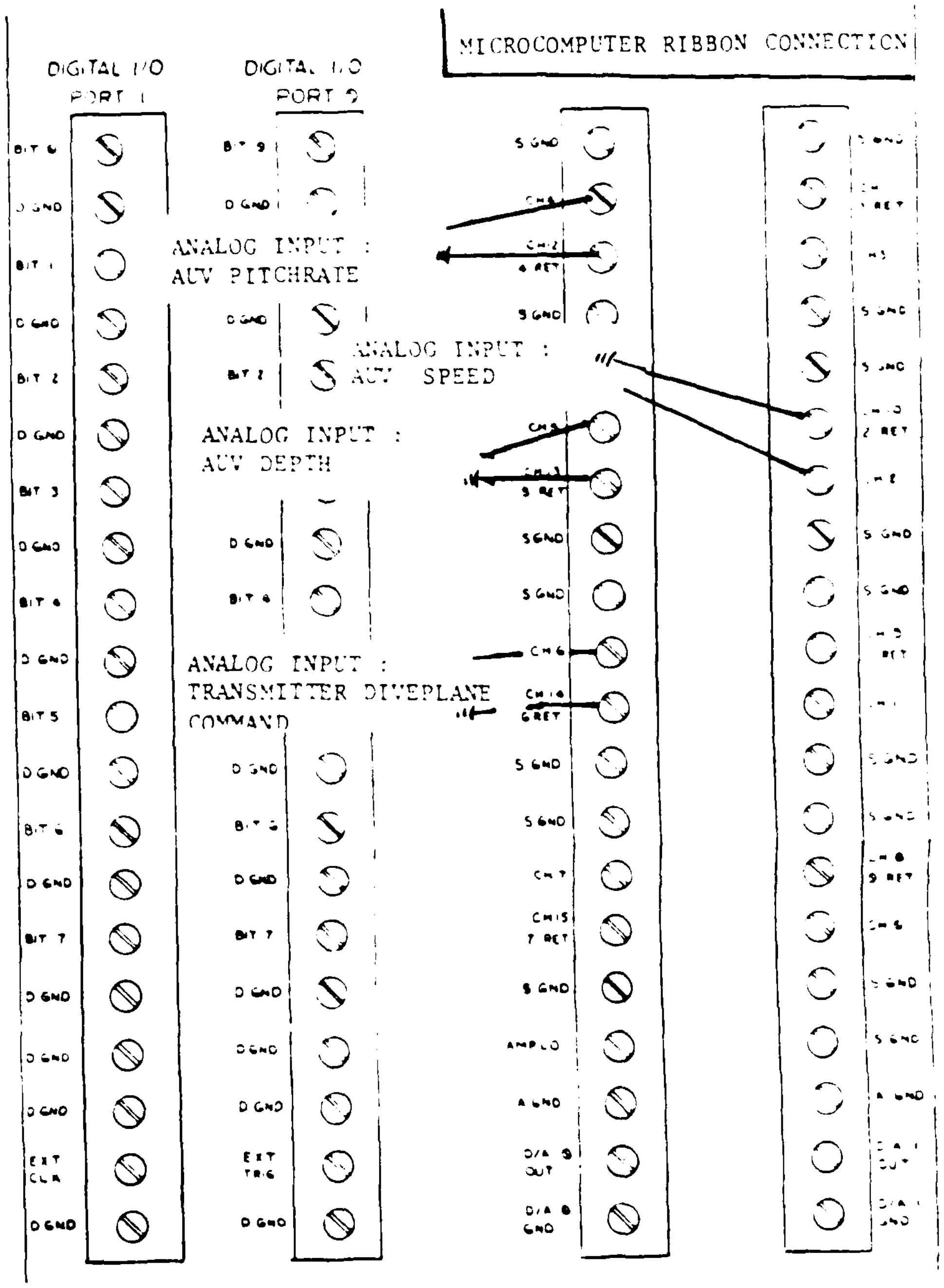


Figure 2.3 Terminal Board Configuration for Data Acquisition

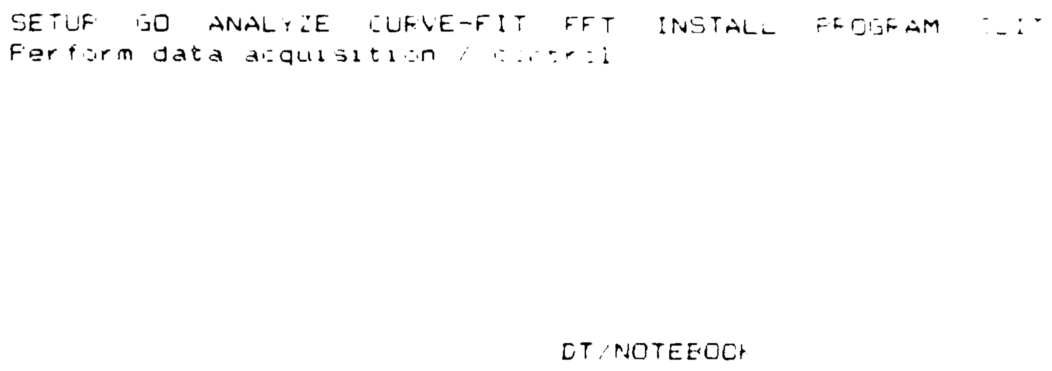

SETUP GO ANALYZE CURVE-FIT FFT INSTALL PROGRAM QUIT
Perform data acquisition / control
DT/NOTEBOOK

Figure 2.4 DT/NOTEBOOK Main Menu

Menu of selections. The Master Key disk may then be removed and is not needed again during the current program session.

The QUIT command exits the program and returns to the Disk Operating System (DOS). The PROGRAM command allows the user to depart the Notebook program and execute programming in DOS and then return to Notebook without having to "re-boot" the program with the Master Key disk. This feature operates must like a programmable interrupt in that the "status" of Notebook is saved for the period that the user is in DOS and is restored upon returning to the Notebook program.

The CURVE FIT and FFT commands are intrinsic program capabilities for curve fitting and fast fourier transform operations on data files. The ANALYSIS command permits the user to call an external analysis program of the user's choosing. (Lotus 123 was used as an example in the documentation.) This permits analysis and graphical representation from within the Notebook program. These features were not used in this research. Acquired data were analyzed and graphed using MATRIXx as a stand-alone program.

The remainder of this sub-section describes the data acquisition process used extensively in this research and which was executed using the SETUP and GO commands from the Main Menu. Figure 2.5 depicts a flowchart of the system configuration "setup" and data acquisition initiation, or "Go," process. It should be noted that once a suitable configuration has been achieved it can be saved as a setup file and recalled for future use. There were several such files saves in this research, however the file named AUVSETUP was used almost exclusively and its configuration is reflected in the figures referenced in the sub-section.

From the Main Menu, selecting the SETUP command results in the presentation of five configuration commands shown in Figure 2.6.

The CHANNELS command allows the user to configure the software to the physical hardware connections of the DT 2801-A/DT 707 interface. The user is first asked to choose between NORMAL or HIGH SPEED data acquisition. The NORMAL command was used throughout this research. The HIGH SPEED option was not needed for the sampling required in this research. This mode places restrictions on data display,

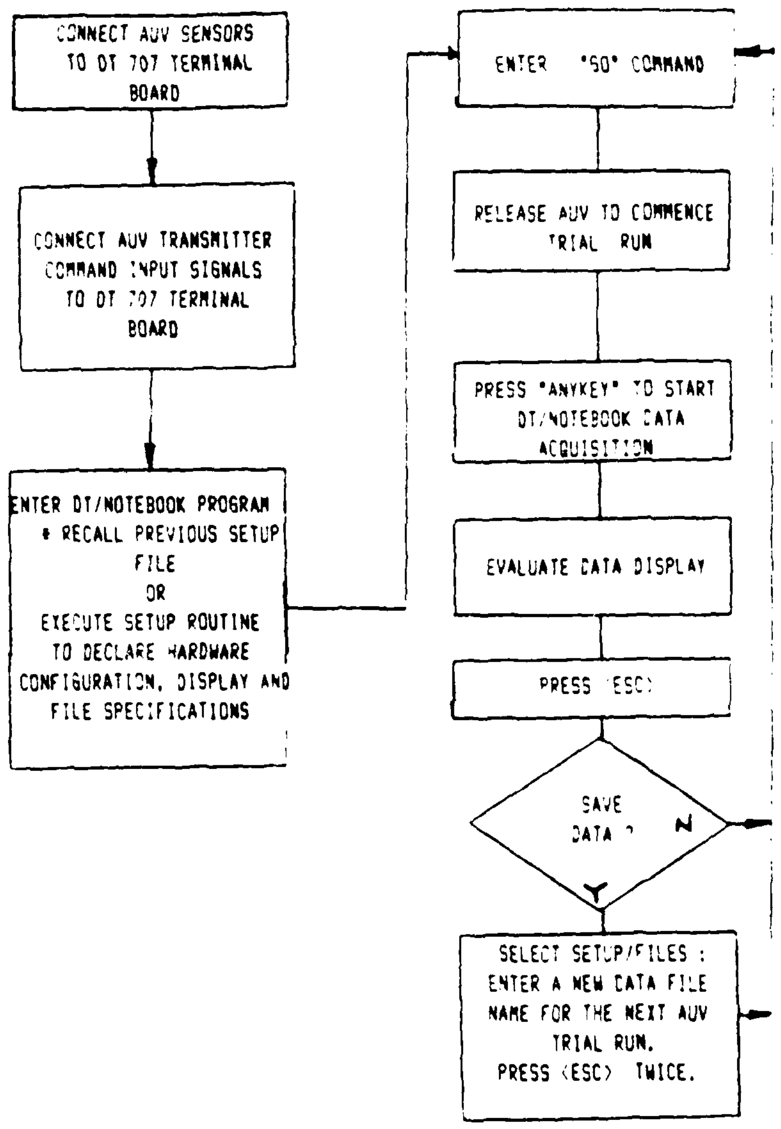
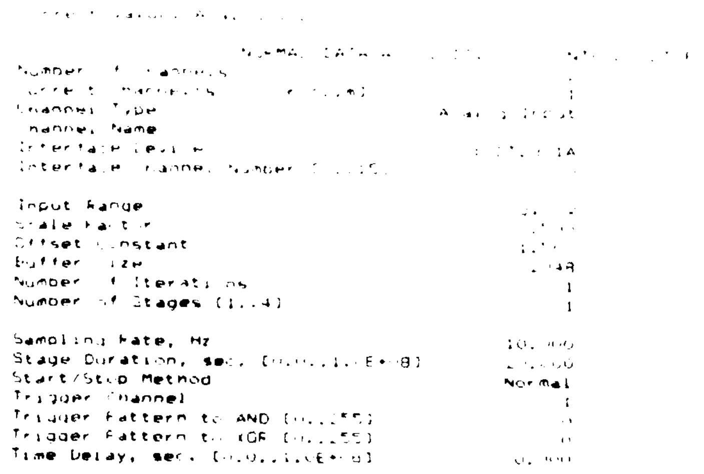

CONNECT AUV SENSORS TO DT 707 TERMINAL BOARD
CONNECT AUV TRANSMITTER COMMAND INPUT SIGNALS TO DT 707 TERMINAL BOARD
ENTER DT/NOTEBOOK PROGRAM
* RECALL PREVIOUS SETUP FILE
OR
EXECUTE SETUP ROUTINE TO DECLARE HARDWARE CONFIGURATION, DISPLAY AND FILE SPECIFICATIONS
ENTER "GO" COMMAND
RELEASE AUV TO COMMENCE TRIAL RUN
PRESS "ANYKEY" TO START DT/NOTEBOOK DATA ACQUISITION
EVALUATE DATA DISPLAY
PRESS <ESC>
SAVE DATA ?
N
Y
SELECT SETUP/FILES : ENTER A NEW DATA FILE NAME FOR THE NEXT AUV TRIAL RUN. PRESS <ESC> TWICE.

Figure 2.5 DT/NOTEBOOK Data Acquisition Process Flowchart

```
CHANNELS  FILES  DISPLAY  VERIFY  SAVE/RECALL
Initialize data acquisition / control
```

Figure 2.6 DT/NOTEBOOK SETUP Command Menu



```
[illegible]

                    NORMAL DATA [illegible]          [illegible]
Number of Channels                                         [illegible]
Current Channel(s)         [1..Num]                          1
Channel Type                                      Analog Input
Channel Name
Interface Device                                      [illegible]A
Interface Channel Number [0..15]                           [illegible]

Input Range                                                [illegible]
Scale Factor                                               [illegible]
Offset Constant                                            [illegible]
Buffer Size                                               2048
Number of Iterations                                          1
Number of Stages [1..4]                                       1

Sampling Rate, Hz                                       10,000
Stage Duration, sec. [0.0..1.0E+8]                     [illegible]
Start/Stop Method                                       Normal
Trigger Channel                                               1
Trigger Pattern to AND [0..255]                               0
Trigger Pattern to OR [0..255]                                0
Time Delay, sec. [0.0..1.0E+8]                           0.000
```

Figure 2.7 DT/NOTEBOOK NORMAL Mode Menu

memory requirements and data buffering and file creation which must be considered if this mode is used.

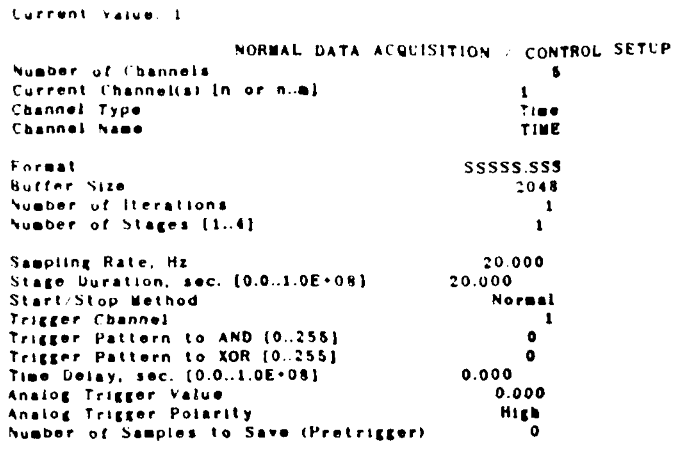
Selecting the NORMAL command results in the presentation of the NORMAL Mode Menu, Figure 2.7. A list of "setup conditions" is presented down the left side with a column of cells for user entries to the right. Entries are made by using the cursor to "capture" the desired cell with the highlighted input cell. Configuration specifications are input using appropriate keys which are "echoed" in the Current Value cell in the upper left hand corner of the menu display. Pressing ENTER or moving the input cell with the cursor key enters the configuration in the program. The program offers some specific options for various setup conditions which are displayed by pressing F1. It should be noted that the DT 2801-A can sample up to 16 input channels (0 ... 15) and these are specified by Interface Channel Number. The user must recognize the distinction between the "data channels" specified in the first four lines of the menu and the interface channel. The data channels are user creations based upon the data requirements. The interface channels are physical input connections to the signal processor.

In this research, data channel 1 was always configured as TIME and used to record elapsed time for an event from the DT 2801-A processor clock. The remaining 4 data channels were specified to receive sensor input signals.

Figures 2.8-2.12 illustrate these configuration specifications.

The next SETUP command is FILES. The option data files are configured to meet user requirements. Figure 2.13 illustrates a data file setup which was used for most of this research. There are two points which need to be emphasized. The first is that after each run, the user must return to this menu and change the "Data File Name." If this is not done, NOTEBOOK will overwrite the new data into the same file. Secondly, the data field specification must be specified to meet the input data file requirements of MATRIXx if the acquired data are to be analyzed in this program. A machine executable data conversion program (convdata.exe) was written to convert NOTEBOOK data into MATRIXx acceptable data. After gaining more familiarity with both these programs, it was discovered that the data conversion program was not necessary. If properly specified in the NOTEBOOK and MATRIXx, data can be used directly. Future researchers should consider the experimental data requirements and then consult the documentation of these programs to effect the proper configuration.

One last documentation remark has to do with the file naming convention used in this research. Conforming to the MS-DOW rules for naming files, data files for AUV trials were named as the following example illustrates:



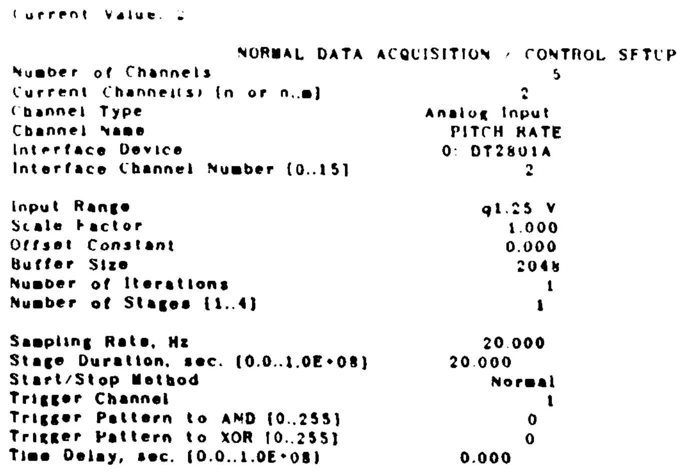
```
Current Value: 1

                         NORMAL DATA ACQUISITION / CONTROL SETUP
Number of Channels                                     5
Current Channel(s) [n or n..m]                         1
Channel Type                                        Time
Channel Name                                        TIME

Format                                         SSSSS.SSS
Buffer Size                                         2048
Number of Iterations                                   1
Number of Stages [1..4]                                1

Sampling Rate, Hz                                 20.000
Stage Duration, sec. [0.0..1.0E+08]            20.000
Start/Stop Method                                 Normal
Trigger Channel                                        1
Trigger Pattern to AND [0..255]                    0
Trigger Pattern to XOR [0..255]                    0
Time Delay, sec. [0.0..1.0E+08]             0.000
Analog Trigger Value                               0.000
Analog Trigger Polarity                             High
Number of Samples to Save (Pretrigger)             0
```

Figure 2.8 DT/NOTEBOOK Channel 1 Configuration



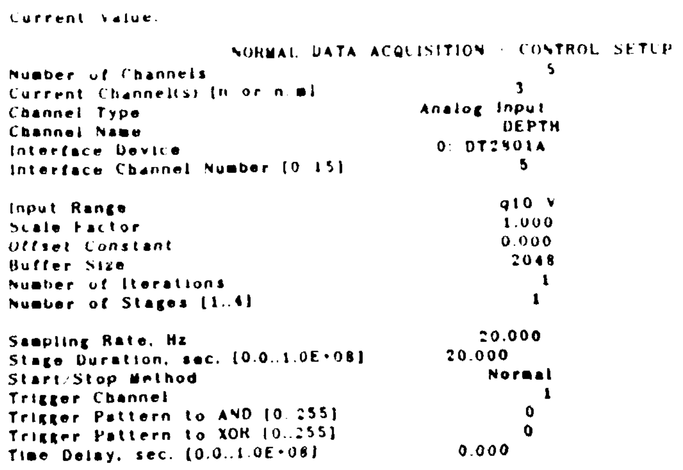
```
Current Value: 2

                         NORMAL DATA ACQUISITION / CONTROL SETUP
Number of Channels                                     5
Current Channel(s) [n or n..m]                        2
Channel Type                                Analog Input
Channel Name                                  PITCH RATE
Interface Device                             0: DT2801A
Interface Channel Number [0..15]                     2

Input Range                                     q1.25 V
Scale Factor                                       1.000
Offset Constant                                    0.000
Buffer Size                                         2048
Number of Iterations                                   1
Number of Stages [1..4]                                1

Sampling Rate, Hz                                 20.000
Stage Duration, sec. [0.0..1.0E+08]            20.000
Start/Stop Method                                 Normal
Trigger Channel                                        1
Trigger Pattern to AND [0..255]                    0
Trigger Pattern to XOR [0..255]                    0
Time Delay, sec. [0.0..1.0E+08]             0.000
```

Figure 2.9 DT/NOTEBOOK Channel 2 Configuration



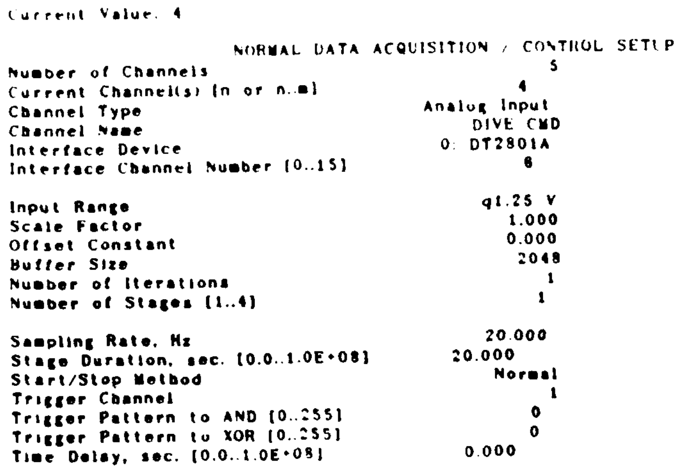
```
Current Value.

                            NORMAL DATA ACQUISITION / CONTROL SETUP
Number of Channels                                            5
Current Channel(s) [n or n.m]                               3
Channel Type                                     Analog Input
Channel Name                                            DEPTH
Interface Device                                  0: DT2801A
Interface Channel Number [0 15]                             5

Input Range                                            q10 V
Scale Factor                                           1.000
Offset Constant                                        0.000
Buffer Size                                             2048
Number of Iterations                                       1
Number of Stages [1..4]                                    1

Sampling Rate, Hz                                     20.000
Stage Duration, sec. [0.0..1.0E+08]                 20.000
Start/Stop Method                                     Normal
Trigger Channel                                            1
Trigger Pattern to AND [0..255]                           0
Trigger Pattern to XOR [0..255]                           0
Time Delay, sec. [0.0..1.0E+08]                     0.000
```

Figure 2.10 DT/NOTEBOOK Channel 3 Configuration



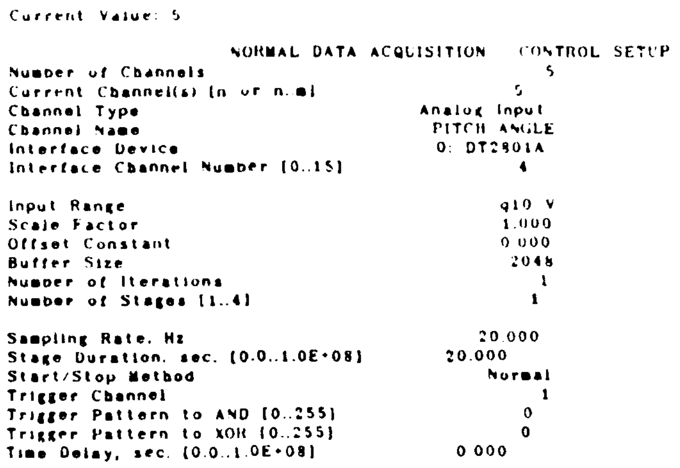
```
Current Value. 4

                            NORMAL DATA ACQUISITION / CONTROL SETUP
Number of Channels                                            5
Current Channel(s) [n or n..m]                              4
Channel Type                                     Analog Input
Channel Name                                         DIVE CMD
Interface Device                                  0: DT2801A
Interface Channel Number [0..15]                            6

Input Range                                           q1.25 V
Scale Factor                                           1.000
Offset Constant                                        0.000
Buffer Size                                             2048
Number of Iterations                                       1
Number of Stages [1..4]                                    1

Sampling Rate, Hz                                     20.000
Stage Duration, sec. [0.0..1.0E+08]                 20.000
Start/Stop Method                                     Normal
Trigger Channel                                            1
Trigger Pattern to AND [0..255]                           0
Trigger Pattern to XOR [0..255]                           0
Time Delay, sec. [0.0..1.0E+08]                     0.000
```

Figure 2.11 DT/NOTEBOOK Channel 4 Configuration



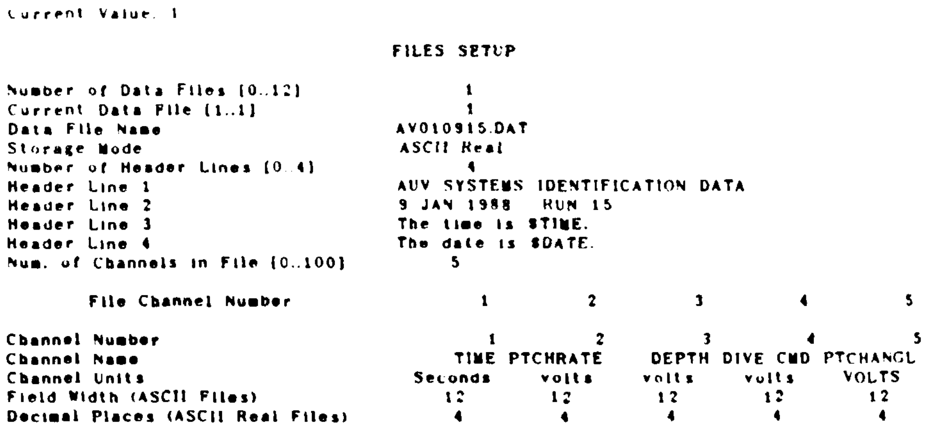
```
Current Value: 5

                         NORMAL DATA ACQUISITION   CONTROL SETUP
Number of Channels                                     5
Current Channel(s) [n or n..m]                         5
Channel Type                                      Analog Input
Channel Name                                       PITCH ANGLE
Interface Device                                  0: DT2801A
Interface Channel Number [0..15]                       4

Input Range                                          ±10 V
Scale Factor                                         1.000
Offset Constant                                      0.000
Buffer Size                                           2048
Number of Iterations                                     1
Number of Stages [1..4]                                  1

Sampling Rate, Hz                                   20.000
Stage Duration, sec. [0.0..1.0E+08]              20.000
Start/Stop Method                                   Normal
Trigger Channel                                          1
Trigger Pattern to AND [0..255]                         0
Trigger Pattern to XOR [0..255]                         0
Time Delay, sec. [0.0..1.0E+08]                  0.000
```

Figure 2.12 DT/NOTEBOOK Channel 5 Configuration



```
Current Value: 1

                                  FILES SETUP

Number of Data Files [0..12]             1
Current Data File [1..1]                 1
Data File Name                    AV010915.DAT
Storage Mode                      ASCII Real
Number of Header Lines [0..4]            4
Header Line 1                     AUV SYSTEMS IDENTIFICATION DATA
Header Line 2                     9 JAN 1988   RUN 15
Header Line 3                     The time is $TIME.
Header Line 4                     The date is $DATE.
Num. of Channels in File [0..100]       5

        File Channel Number              1          2          3          4          5

Channel Number                           1          2          3          4          5
Channel Name                           TIME   PTCHRATE      DEPTH   DIVE CMD   PTCHANGL
Channel Units                       Seconds      volts      volts      volts      VOLTS
Field Width (ASCII Files)               12         12         12         12         12
Decimal Places (ASCII Real Files)        4          4          4          4          4
```

Figure 2.13 DT/NOTEBOOK Data Files Setup

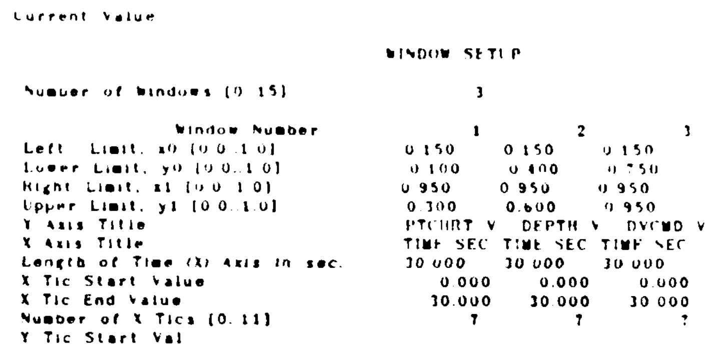
AV010915.DAT

AV :	denotes AUV data acquired from NOTEBOOK
01 :	denotes the month of the trial, i.e., January
09 :	denotes the day of the month of the trial
15 :	denotes the particular trial for the day
DAT:	was the extension used for all data files.

Files which were converted by the data conversion program for analysis with MATRIXx were named by the same convention except "UV" was used for the first two characters. Thus the file UV010915.DAT would indicate that this is the data file converted from AV010915.DAT for use with MATRIXx.

The next SETUP menu option is DISPLAY. This command specifies the setup for the real-time display of data as it is being acquired. This feature is most beneficial in evaluating the quality of the data collected. It should be noted that in achieving a visual display the program invokes a liberal "graphical license" and so the displays afford a relative measure and not an exact replication of the data acquired. The stored data file is unaltered by the display configuration specification.

The DISPLAY command presents the user with two menus: WINDOW SETUP and TRACE SETUP. In these menus the user can specify the number, size and type of the display windows and the data trace characteristics. There is considerable flexibility and the program documentation should be consulted to exploit this feature. Figures 2.14 and 2.1 illustrate configurations used in this research to display three windows and their respective traces.



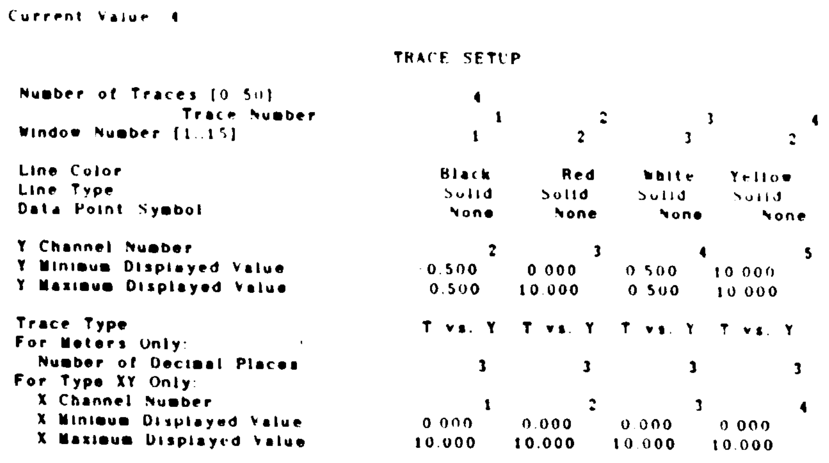
```
Current Value

                                        WINDOW SETUP

Number of Windows [0 15]                        3

              Window Number                     1          2          3
Left  Limit, x0 [0.0..1.0]              0.150      0.150      0.150
Lower Limit, y0 [0.0..1.0]              0.100      0.400      0.750
Right Limit, x1 [0.0..1.0]              0.950      0.950      0.950
Upper Limit, y1 [0.0..1.0]              0.300      0.600      0.950
Y Axis Title                            PTCHRT V   DEPTH V    DVCMD V
X Axis Title                            TIME SEC   TIME SEC   TIME SEC
Length of Time (X) Axis in sec.         30.000     30.000     30.000
X Tic Start Value                          0.000      0.000      0.000
X Tic End Value                           30.000     30.000     30.000
Number of X Tics [0..11]                    7          7          7
Y Tic Start Val
```

Figure 2.14 DT/NOTEBOOK Window Setup Configuration



```
Current Value 4

                                   TRACE SETUP

Number of Traces [0 50]                      4
                 Trace Number                  1          2          3          4
Window Number [1..15]                        1          2          3          2

Line Color                                Black       Red      White     Yellow
Line Type                                 Solid     Solid      Solid      Solid
Data Point Symbol                          None      None       None       None

Y Channel Number                              2          3          4          5
Y Minimum Displayed Value               -0.500     0.000      0.500     10.000
Y Maximum Displayed Value                0.500    10.000      0.500     10.000

Trace Type                              T vs. Y   T vs. Y    T vs. Y    T vs. Y
For Meters Only:
  Number of Decimal Places                    3          3          3          3
For Type XY Only:
  X Channel Number                            1          2          3          4
  X Minimum Displayed Value              0.000     0.000      0.000      0.000
  X Maximum Displayed Value             10.000    10.000     10.000     10.000
```

Figure 2.15 DT/NOTEBOOK Trace Setup Configuration

The next SETUP is VERIFY. Executing this command displays a screen which reflects the configuration which a user has specified. This provides a good check of the system setup.

The last SETUP command is SAVE/RECALL. As the title suggests this command allows the user to save a particular configuration setup as a specific file or to recall such a file for reuse.

Once a system configuration has been specified and the AUV is connected to the DT 2801-A/DT 707 interface, actual data acquisition is initiated by selecting the GO command from the Main Menu. In its default configuration, the program will commence data acquisition as soon as GO is selected. However, a useful feature is the "Keystroke Before Run" which is selectable in the INSTALL, OPTIONS commands. This delays the beginning of data acquisition after GO is selected until "any other key" is depressed. This feature was used throughout this research as it permitted more precise control over the data acquisition events.

F. DOCUMENTATION AND CONFIGURATION FOR AUV CONTROL

Figure 2.16 depicts the system configuration for control of the AUV under high-level language control programming. Figure 2.17 shows a schematic of the DT 707 terminal connections. Owing to the non-availability of the test tank and suitable radio transmitter interface module to transmit

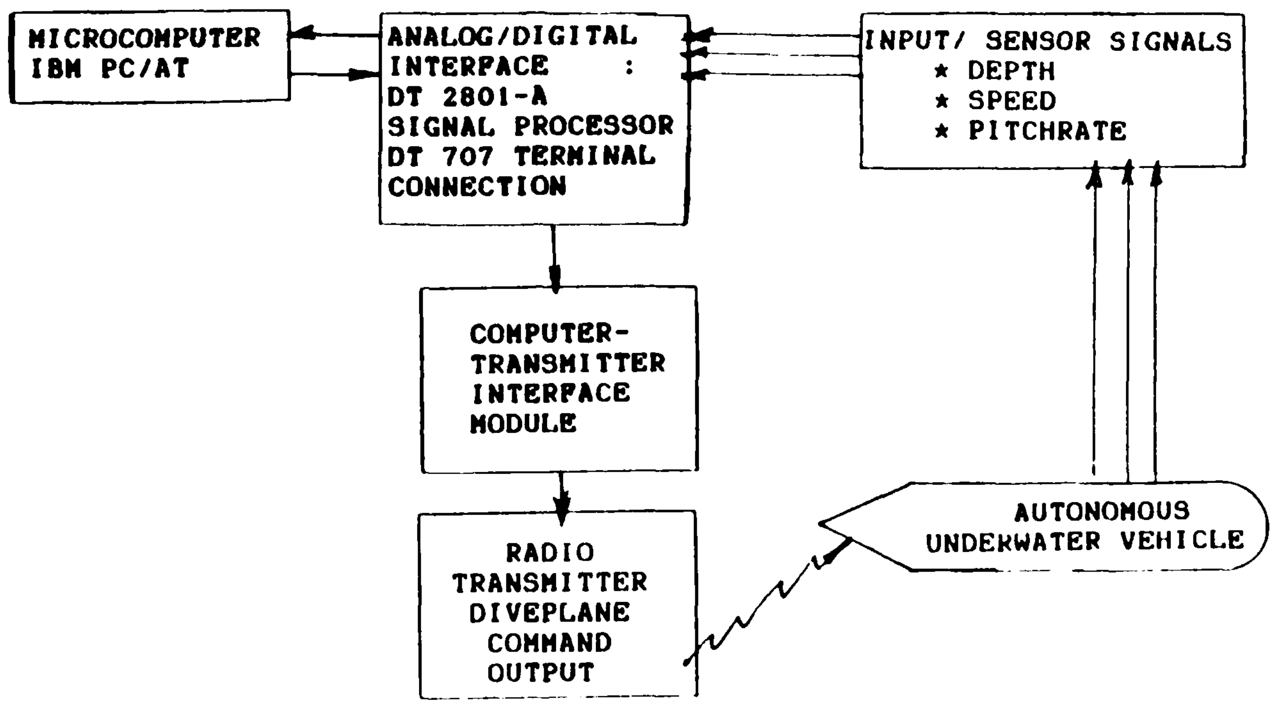

MICROCOMPUTER
IBM PC/AT
ANALOG/DIGITAL INTERFACE : DT 2801-A SIGNAL PROCESSOR DT 707 TERMINAL CONNECTION
INPUT/ SENSOR SIGNALS
* DEPTH
* SPEED
* PITCHRATE
COMPUTER-TRANSMITTER INTERFACE MODULE
RADIO TRANSMITTER DIVEPLANE COMMAND OUTPUT
AUTONOMOUS UNDERWATER VEHICLE

Figure 2.16 System Configuration for Real-time AUV Control

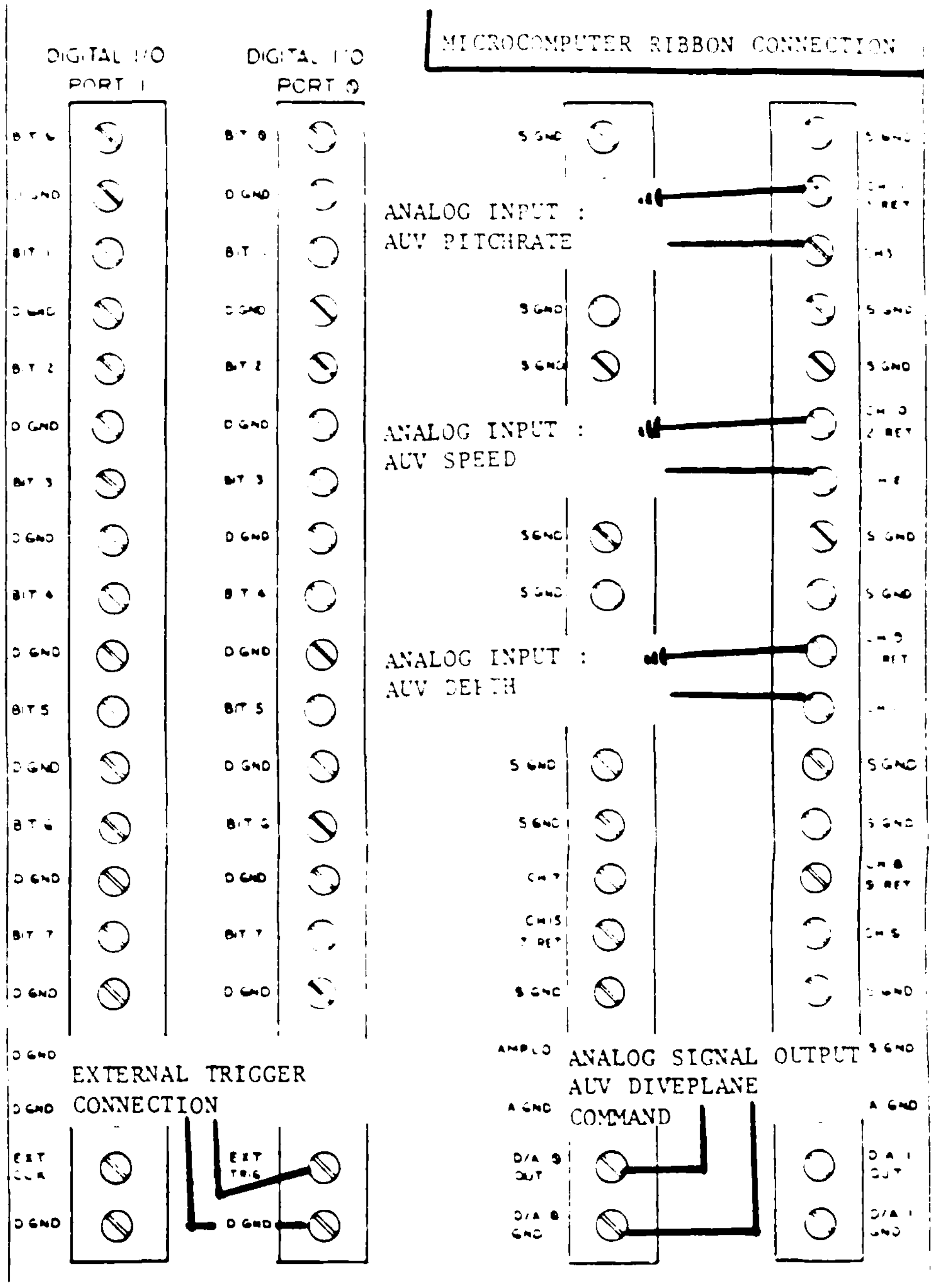


Figure 2.17 Terminal Board Configuration for Real-time Control

the generated dive plane command to the AUV, real-time control was not accomplished in this research. Figure 2.18 shows the system configuration used to simulate real-time control configuration in the development and verification of the control program. Details of the program development and verification are reported in Chapter III of this thesis.

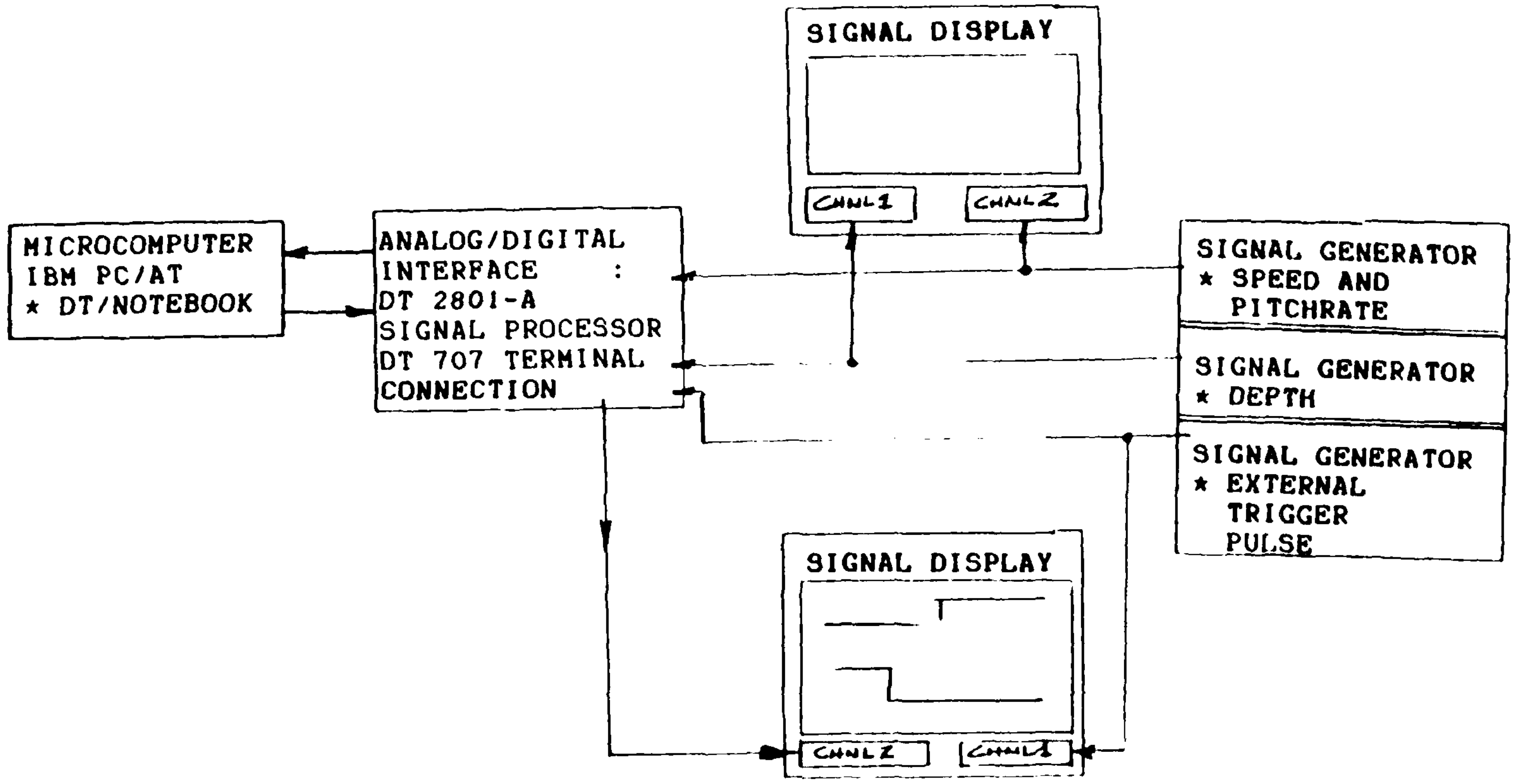


Figure 2.18 System Configuration for Simulation of Real-time AUV Control

III. PROGRAMMING VERIFICATION

This chapter reports on the final objective of this thesis: verification of I/O processing of analog and digital signals under higher-level program control. The following sections discuss the design, implementation, verification and documentation of this program. A complete and documented listing of the main program is contained the Appendix.

A. PROGRAM PERSPECTIVE AND DESIGN

The introduction of [Ref. 2] presents a concise description and illustration relating this concept to this and other current research at the Naval Postgraduate School. The theory of classical (closed loop) controllers as well as modern state-space design of Digital Autopilots are further discussed in [Refs. 10,11]. In its final implementation, a digital autopilot will control all six degrees of freedom of the AUV in the execution of a variety of missions and in a diversity of operating environments. The architecture of the autopilot will reflect extensive functional modularity. Under the supervisory control of an "AUV Operating System," missions will be executed as a sequence of tasks compiled and ordered by the onboard planning logic and knowledge database. Similarly, completion of these tasks will reflect the compilation and execution of functional capabilities in sequence or in parallel under the control of a lower-level operating system. These functional capabilities may well be implemented as multi-processors each reflecting a hierarchical organization specific to their required functional responsibility.

Timing will be a critical to the successful assimilation of functional modules and the accomplishment of tasks and missions. The top level "AUV Operating System" will provide a master synchronizing timing signal based optimal considerations of the cycle-time requirements of the subordinate processors. Cycle-time is the time required to receive, process and transmit information. Cycle-time will be most affected by the processing interval. Modern digital controller design must therefore concern itself with optimizing response and processing time requirements.

A Model-Referenced Autopilot design was evaluated and found to be most suited to control of the rapid maneuvering and changing environmental and "plant" parameters which will be encountered in an AUV [Ref. 2] In contrast to the classical accomplishment of vehicle maneuvers through the independent action of rudder and diveplane for course and depth control, a Model Referenced design will be a multivariable control structure. It will require the parallel processing of vehicle sensory data to determine a vehicle state, comparison with a model reference state and the determination of an appropriate command. As verification of the on-site program development capability, this research sought to investigate the cycle time requirements of a high-level language control program which emulated the modular organization of a model referenced controller and exercised the signal processing functions of the DT 2801-A hardware and software.

B. PROGRAM IMPLEMENTATION

This program represents a functional module at the lowest level of the digital autopilot hierarchy: the analog-digital interface between the AUV servo-controllers and the onboard digital processing capability. As discussed in Chapter II, the verification process was simulated using signal generators (Figure 2.18). Three analog sensory signals, simulated by inputs from signal generators are sampled. These signals, represent AUV transducer outputs for depth, speed and pitchrate. The amplitude of these signals are representative of actual AUV signals measured during tank testing. Respective ranges of amplitudes of these signals are +/- 10, +/- 5, and +/- 1.25 volts. The sampled analog signals are then processed by the program: the speed and pitchrate signals are processed for display only; the depth signal is processed by the control modules which generate an outgoing command (voltage) to the diveplane actuator channel. An external trigger simulates the master timing signal as it might be implemented in an actual processor onboard an AUV.

The program implements a closed loop control process within a user interface shell. As depicted in the program pseudo-flow chart (Figure 3.1), the user enters a desired AUV target depth and then turns control of the AUV over to the control loop. It was intended that the user have the facility to interrupt the control sequence by pressing key <F1> to enter a new target depth, <F2> to halt the controller and reset the program, or <ESC> to exit the program. It was discovered in the final assembly of the program modules, however, that the PCLAB signal processor routines masked the keyboard interrupt device thus denying access to programmable interrupts for program control. Although this problem did limit the interactive features of the program, it did not interfere with the active control or validation of the development system. This problem is discussed in Chapter IV.

This program utilizes routines which are provided with the PCLAB real-time software. In many of the compiled language implementations which includes Turbo Pascal,these routines are programmed as functions which return integer error codes as well as passing analog or digital data values. Programming these routines is accomplished by declaring an integer variable and then calling the desired routine. For example, STATUS is the declared integer

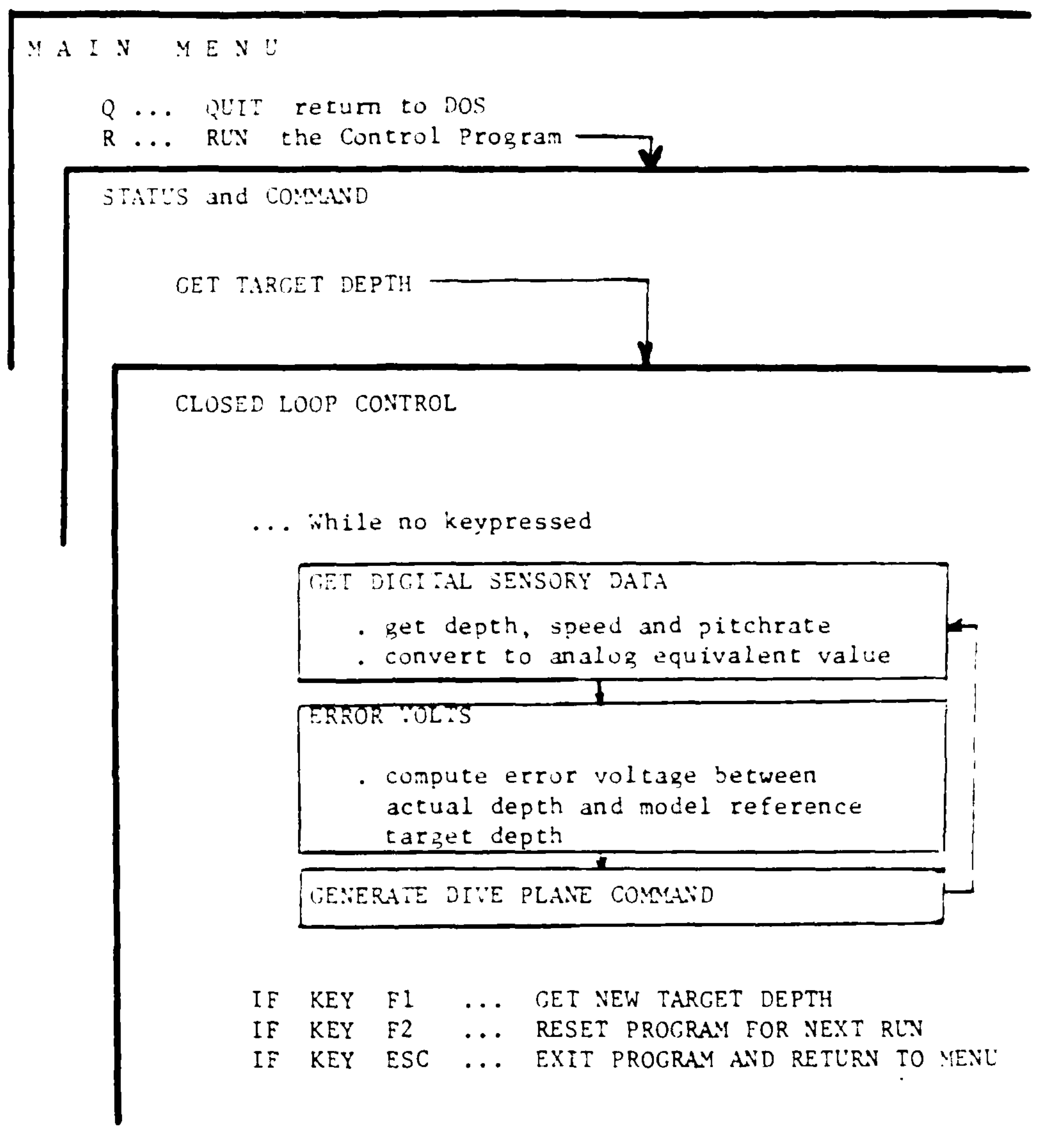

M A I N   M E N U
Q ... QUIT return to DOS
R ... RUN the Control Program
STATUS and COMMAND
GET TARGET DEPTH
CLOSED LOOP CONTROL
... While no keypressed
GET DIGITAL SENSORY DATA
. get depth, speed and pitchrate
. convert to analog equivalent value
ERROR VOLTS
. compute error voltage between actual depth and model reference target depth
GENERATE DIVE PLANE COMMAND
IF KEY F1 ... GET NEW TARGET DEPTH
IF KEY F2 ... RESET PROGRAM FOR NEXT RUN
IF KEY ESC ... EXIT PROGRAM AND RETURN TO MENU

Figure 3.1 Closed Loop Control Program Flow Chart

variable used in the program code. Calling a single value Analog to Digital conversion is accomplished with the following statement:

```
  status= adcvalue (<channel ¯,<gain>,<user declared
                                       data variable >)

    <   > denote required declarations
```

Literally interpreted, this statement says that "Status gets the integer error code of the function adcvalue." Error codes are automatically processed by an error processor and if an error is encountered in the data conversion, an error message is printed to the display screen. The data values are passed to the program as value parameters and may be treated as any other data parameter in program coding and manipulation. PCLAB reportedly supports single or block data conversion routines. However several attempts to use the block routines to convert data from selected sensor channels were not successful. This problem is discussed in Chapter IV.

The following sub-sections supplement the program commentary found in program listing (Appendix), and describe the salient modules which implement the Closed Loop Control routine. The titles of the subsections correspond to the title of the program modules as listed in the Appendix. Appendix D [Ref. 5] contains the algorithms for analog and digital conversions. These have been used in the following procedures where such data manipulation is required.

1. Initialize Zero Digital Signal Out

This routine must executed at the beginning of the program so as to send a zero voltage digital signal out. The DT 2801-A defaults to sending out a minimum full scale reading out as soon as it is powered up. It is therefore important to "zero" the output signal before attaching the test vehicle. This routine "zeroes" the signals of both analog to digital output channels.

2. Convertspeed, Convertdepth and Convertpitchrate

These are three functions which convert digital-to-analog data values to AUV parameters. The algorithms for these conversions are based on tank tests conducted on the AUV and are reported in [Ref. 3]

3. GetTargetDepth

This procedure solicits the AUV operating depth from the user (an integer input) and converts it to an analog voltage based upon the depth to voltage relationship derived during testing. This equivalent analog voltage is passed to the control program for control processing.

4. GetDigitalSensoryData

This procedure samples three channels of sensory input and converts them to AUV analog equivalent values using the previously mentioned functions. These values are then passed to the control program as representing the state of the AUV. Several designs were envisioned using single and block data transfers so as to achieve a suitable throughput and synchronization of this event. ADConTrigger is a single value conversion routine which executes on the high to low transition of an external trigger. As soon as this conversion is completed, two ADCValue commands convert data for the remaining two sensory channels. Timing studies for a single value conversion affirmed a throughput on the average of 0.6 msec. The throughput for the one triggered and two single value conversions averaged 2.5 msec.

5. Errorvolts

This procedure computes a difference or error voltage between the analog voltages for target and actual depths. A "voltage filter" has been implemented to permit a tolerable difference range above and below the target depth. This precludes trying to achieve an absolute zero difference which is an not a practicable design in manipulating the digital to analog conversions. It also allows the AUV a defined range in which no changes in plant operating parameters are required to maintain a desired depth. The computed error voltage is then passed to the control program for processing by the GenerateDiveCommand module.

6. GenerateDiveCommand

This procedure simply converts the computed error or difference voltage to an analog equivalent and sends this signal out on the specified analog to digital channel.

C. PROGRAM VERIFICATION

The verification configuration and objectives have been previously stated. The evaluation of the cycle time requirement is based on the minimum required AUV sampling time in order to maintain real-time control. A critical real-time control interval of 50 mSec or a sampling rate of 20 Hz was established as a standard based upon evaluation of data collected and reported in [Ref. 3].

Notwithstanding the problems encountered with programmable interrupt control and block data conversions using the PCLAB routines, high-level program control of the analog and digital I/O signal processing was achieved. Cycle times were measured at various sampling rates as simulated by the frequency of the external trigger signal. Figures 3.2-3.4 show the results of these timing studies. A cycle time on the order of 5 msec was recorded. This cycle time is well within the required limit and affords an ample margin for more extensive sensor sampling, control processing and multiple command generation.

D. PROGRAM DOCUMENTATION

The documentation of the program listing and the previous sections in this chapter have described the logic and function of the program. This section documents the filenames and other information which may assist follow-on research.

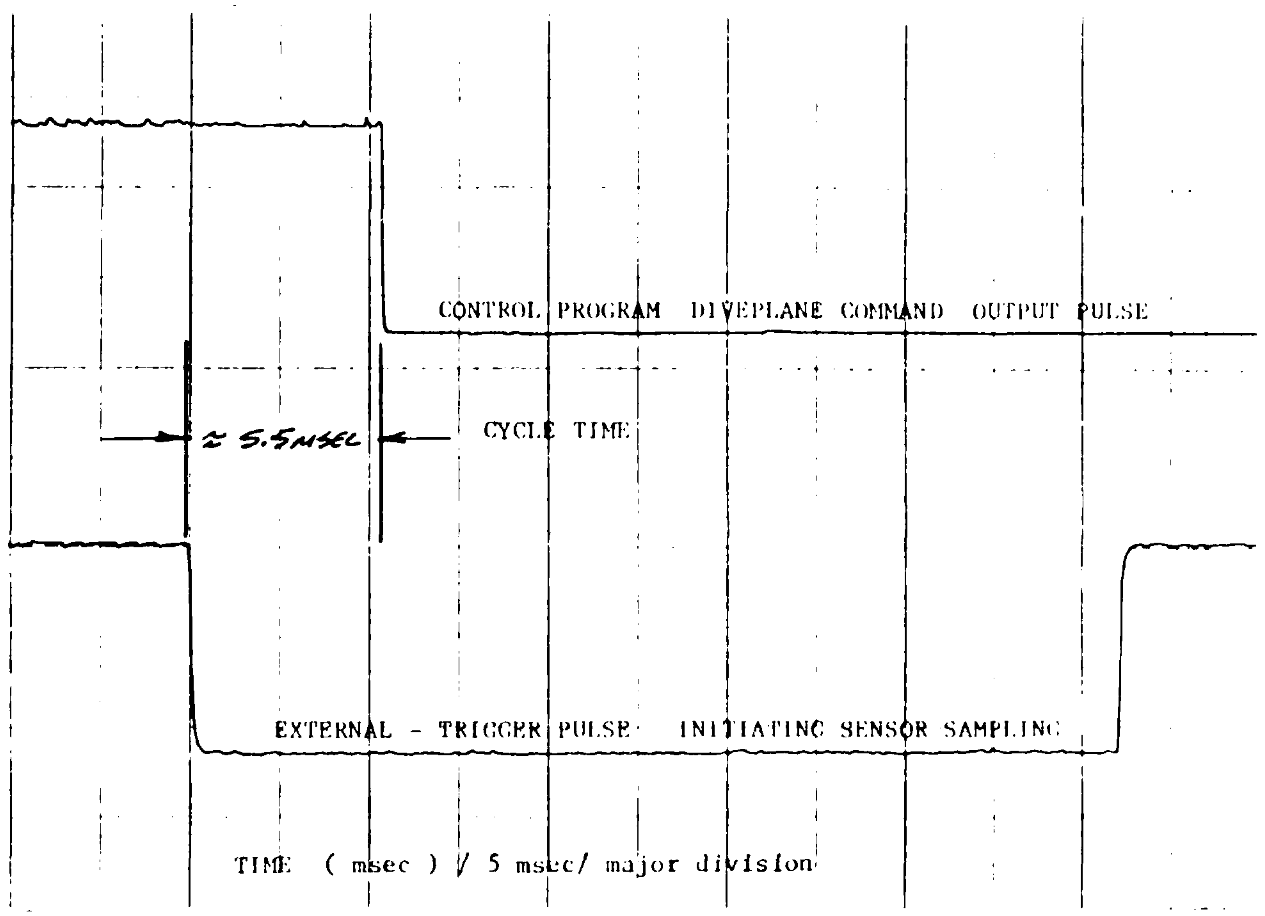


Figure 3.2 Program Cycle Time Trace at 8 Hz Trigger Rate

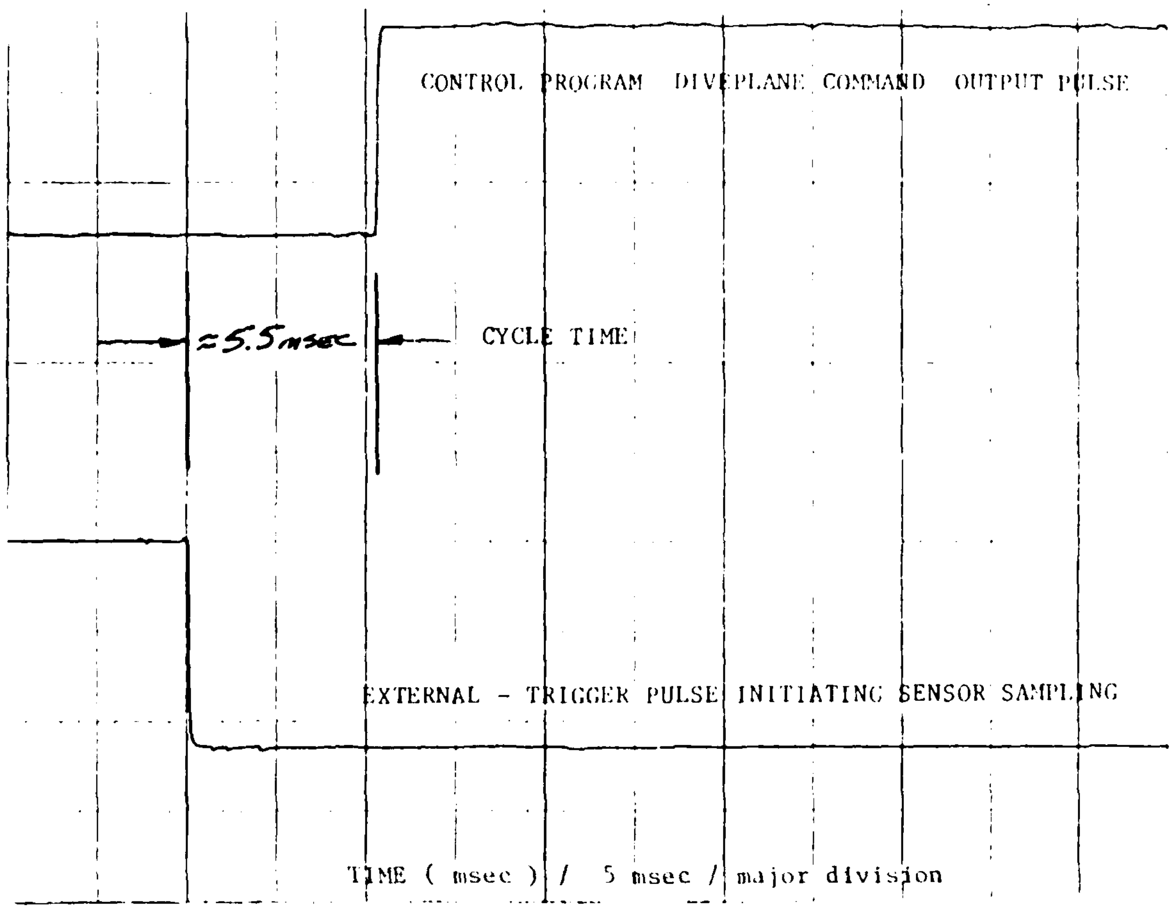


Figure 3.3 Program Cycle Time Trace at 12 Hz Trigger Rate

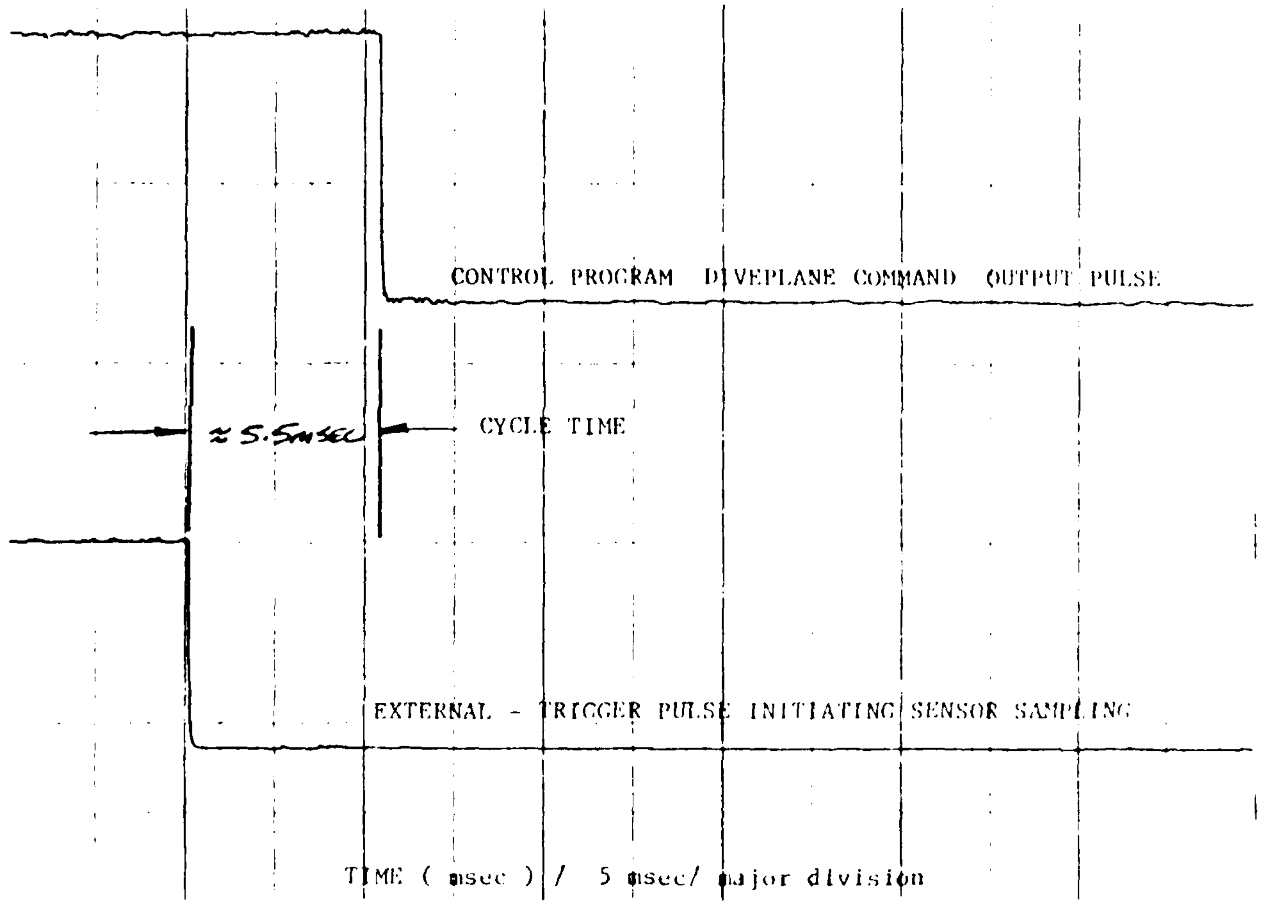


Figure 3.4 Program Cycle Time Trace at 20 Hz Trigger Rate

The control program is named AUVPILOT with extensions of .PAS and .COM for the Turbo Pascal and executable machine codes, respectively. The included files declared at the beginning of the program listing (Appendix) all have a file extension .AUV. These programs provide utility support for screen displays and program timing. Some of these programs are adaptations of routines discussed in [Ref. 9], while others were created to meet specific needs of the main program.

The Turbo Pascal v 3.0 programming environment and language features are thoroughly explained in [Ref. 8]. No additional explanation is considered necessary. One major consideration in the utility of this programming language is that in the recently released 4.0 version, the code size limitation of 64K bytes has been eliminated thus making it suitable for very large program applications. The compatibility problems with PCLAB routines still remain to be resolved, however. PCLAB does not as yet support this latest version.

IV. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

A. SUMMARY

The thesis has described the hardware and software details of an IBM-PC based controller as a shell for further development. The software has been used for digital data acquisition and cycle time studies have been performed for throughput of control signals operating fr m an external trigger source.

B. CONCLUSIONS

The work of this thesis demonstrated the ability of an IBM-PC based microcomputer, coupled with an advanced D/A and A/D conversion board to provide both data acquisition and control functions that will be required in any implementation of an intelligent controller for Automated Underwater Vehicles.

Cycle time for sensing three signal channel input, computation of tracking error, and sending of updated control signals is 5 msec. With a sample time of 0.05 seconds, 45 msecs are available for storing data, updating vehicle parameter estimates, and interfacing with higher level supervisory control programs. The trigger for sample initiations, at present, is timed by an external clock.

The structure implemented herein, is now capable of being expanded to incorporate the control signal formulations necessary to achieve full implementation of an automated digital autopilot.

C. RECOMMENDATIONS

The problems with programmable interrupt control and multiple, or block, data conversions discussed in Chapter III present several issues which need to be resolved. First, if Turbo Pascal is to be used in future research, these problems will have to be resolved through direct correspondence and consultation with the development firm. Furthermore, the near-term availability of PCLAB routines for Turbo Pascal v 4.0 should be assured so as to support development of large control programs. At this juncture in the research, it is perhaps best to evaluate other compiled high-level languages with respect to the stated benefits of Turbo Pascal and the programming requirements of more complex Digital Autopilot implementations. Program interfacing with other, more "intelligent" digital control modules should also be considered so as to assure future integration of program modules under development by the various departments at the Naval Postgraduate School.

APPENDIX

LISTING OF CONTROL PROGRAM CODE

```
program   AuvAutoPilot  ( input, output );

{ TITLE        : Autonomous Underwater Vehicle Auto Pilot Program.
  AUTHOR       : CDR Stephen W. Delaplane, USN
  APPLICATION : Partial fulfillment of Thesis Research for the degree
                 Masters of Science in Mechanical Engineering.
  DATE         : 18 March 1988

  Project Description : This program implements digital control of the NPS
  autonomous underwater vehicle (AUV) in the vertical or dive plane.it samples
  vehicle sensor input from three channels : depth, speed, pitchrate.  The
  depth signal is then passed to a DepthError module which compares the actual
  sensor depth with a model reference depth simulated by a depthgain.  A depth
  error voltage is then generated and passed to a GenerateDivePlaneCommand
  module which processes the error signal and sends out an appropriate command
  to the diveplane actuators. The gains in the algorithms reflect the discrete
  transfer function gains for diveplane command response derived from vehicle
  identification analysis.     }

{ GLOBAL DECLARATIONS    -------------------------------------------------

const
{ ----------- Screen declarations --------------- }

   x1            = 5;                     { Upper left corner : left edge }
   y1            = 2;                     { Upper left corner : upper edge }
   x2            = 75;                    { Lower right corner : right edge }
   y2            = 24;                    { Lower right corner : bottom edge }

type
  str10 = string [10];
  str60 = string [60];

var
  hr,hr2,min,min2,
  sec,sec2,hun,hun2              : byte;
  seconds                         : real;
  option, controlmode,
  reply,reply2                   : char;

{------------------------- INCLUDED FILES Declarations    ------------------ }


{$I pcldefs.tp  }   { PC LAB Trubo Pascal routines.                              }
{$I pclerrs.pas }   { PC LAB error code messages file.                           }
{$Iinidac.auv    }
{ This procedure initializes the DT 2801-A TO ZERO VOLTS AND MUST BE
EXECUTED  BEFORE THE AUV IS HOOKED TO THE COMPUTER.   It is called as
procedure InitializeZeroDigitalSignalOut.                                   }

{$Igettime.auv   }  {   No arguments; returns hr, min, sec, hun : byte       }

{$Ishotmdff.auv }
{ Input the output of TimeDiff.auv and this procedure displays the time
  difference between the two most current GetTime.auv results.
  ShowTimeDifferene ( x:integer) .                                          }
```

```
{$Itimediff.auv}
{ Input: hr,hr2,min,min2,sec,sec2,hun,hun2 from two calls
  of GetTime.auv and this returns the difference in
  seconds as a REAL variable.                                   }

{$Idrawbox2.auv}
{ Input x1,y1,x2,y2 : integer to specify the corner limits of the box.
This procedure clears screen and draws a rectangular box of specified
dimension using ASCII double line characters.                }

{$Iclrbox2.auv}
{ Input x1,y1,x2,y2 : integer to specify the corner limits of the box.
  This procedure uses a FAST means of clearing a box of specified dimension.
  The box dimension should be delcared as constants.            }

{$Iboxprint.auv }
{ Input the printrow, leftboxedge, rightboxedge  : integer and printstring :
  str60.  This procedure centerprints the string in the box at the printrow
  specified without overwriting the box border.                 }

{$Ishowfast.auv}
{ Input message : str60, column,row : integer.  To specify the x,y position
on the screen for a FAST message print.                         }

{$Ikeyhit.auv}
{ This is a boolean function which returns true or false if key is pressed;
it also returns keycode replies VAR reply, reply2 : char  .     }

{$Itabxy.auv}
{ Input tabcol,tabrow : integer; like gotoxy                    }

{$Iboxpause.auv}
{ Input xpause,ypause : integer to specify where "Press any key to continue"
  message is to be printed.                                     }

{$Igetkey.auv }
{ Input as a string of chars, the set of acceptable replies; ie 'YyNn'.  This
  procedure waits until one of the acceptable replies has been entered.  }

{$Iutils.auv   }
{ Included are some housekeepping and debugging routines.       }

{$Iconvadv.auv }
{ Includes functions to convert depth,speed and pitchrate to vehicle values. }
{ ****************  MAIN PROGRAMS PROCEDURES     ***************************
{ **************** USER INTERFACE MODULES        ***************************
procedure MainMenu ( var reply : char );

{  This procedure presents the AUV screen and solicits an option to Run
   the AUV from the StatusAndCommand procedure or to Quit.                }

begin
   repeat
     clrscr;
     drawbox2(x1,y1,x2,y2);
     boxprint(y1+3,x1,x2,'N A V A L   P O S T G R A D U A T E   S C H O O L');
     boxprint(y1+5,x1,x2,'D E P A R T M E N T   O F ');
     boxprint(y1+6,x1,x2,'M E C H A N I C A L   E N G I N E E R I N G');
     boxprint(y1+8,x1,x2,'AUTONOMOUS   UNDERWATER   VEHICLE');
     boxprint(y1+10,x1,x2,'DIGITAL   AUTOPILOT  CONTROL   PROGRAM');
     boxprint(y1+12,x1,x2,'**************************************');
     boxprint(y1+15,x1,x2,'Do You want to RUN this program ..');
     boxprint(y1+16,x1,x2,'or Do You want to QUIT and return to DOS ?');
     boxprint(y1+20,x1,x2,'>>>>  ENTER Q OR R  <<<<');
     getkey ('QqRr',reply,reply2);
   until ( reply in ['Q','q','R','r'] ) and (reply2 = chr(0));
end;

procedure StatusAndCommand ( var mode : char );
{ This procedure begins the control program  screen.                }

var
   mode2                : char;

procedure StatusAndCommandScreen;
{  This is the status and control screen and solicits a user input of F1 to
   RUN the program or Q to Quit and exit to the main menu.                  }

begin        { --------------- StatusAndCommandScreen ------------------ }
      clrbox2 (x1,y1,x2,y2);
      boxprint(y1+1,x1,x2,'AUV STATUS / COMMAND AND CONTROL SCREEN');
      boxprint(y1+2,x1,x2,'============================================');
      boxprint(y1+7,x1,x2,'CHOOSE YOUR DESIRED CONTROL MODE  :');
      boxprint(y1+9,x1,x2,'ENTER KEY  << F1 >>   TO START AUV CONTROL');
      boxprint(y1+11,x1,x2,'ENTER  << Q >> TO QUIT AND RETURN TO MAIN MENU');
      boxprint(y1+16,x1,x2,'PRESS EITHER   F1   OR  Q');

end;         { ---------------- StatusAndCommandScreen ------------------ }
```

```
{ ********************** CLOSED LOOP CONTROL ROUTINES ********************

procedure ClosedLoopControl;
{ This module comprises the closed loop control scheme.                    }

const
     maxdepth          - 33;
     mindepth          = 0;
     updateincrement   = 10;

type
   activecontrolmode    = ( run, reset, exit );
   allowabledepthrange  = mindepth..maxdepth ;

   auvattitude          = ( climb, maintain, diving );

   digitalintegerarray  = array [1..3] of integer;

var
   auvdepth, auvdepthvolts,auvspeed,
   auvspeedvolts, auvpitchrate, auvpitchratevolts,
   deptherrorvolts, targetdepthvolts                  : real;
   adv                                                : digitalintegerarray;
   j,status                                           : integer;
   modereply,modereply2                               : char;
   activemode                                         : activecontrolmode;
   targetdepth,
   updatecounter                                      : integer;
   depthrange                                         : allowabledepthrange;
   attitude                                           : auvattitude;

function convertdepth ( analogvalue : real ) : real;

{ This function converts a depth analog volts value to an AUV status
  parameter. This function is derived from experimental observation.     }

begin                              { --- function convertdepth  ------------- }

   convertdepth := ( analogvalue - 1.6270 ) / 0.2570 ;

end;                               { --- function convertdepth  ------------- }

function convertspeed ( analogvalue : real ) : real;

{ This function converts a speed analog volts value to an AUV status
  parameter. This function is derived from experimental observation        }

{ Speed of the AUV was determined by hand-timing the vehicle's passage in the
test tank, over a distance of 8ft ( 2 window panels ) while holding the speed
voltage constant. Various trials were performed at different input voltages
to establish a voltage to speed relationship. In theory this is a quadratic
relationship. The data alluded to a quadratic relationsip, but because the
voltage saturated the range quickly, it was difficult extract a precise
function. Accordingly this function asserts average speed values for various
ranges of voltages based on the timing trials. This conversion will
undoubtedly become better defined as more vehicle data is taken and analyzed.}

begin                              { --- function convertspeed  ------------- }
```

```
    if      ( analogvalue <= 2.32 ) then
       convertspeed := 0.0
    else if ( analogvalue > 2.3200 ) and ( analogvalue <= 2.5200 ) then
       convertspeed := 1.2678
    else if ( analogvalue > 2.5200 ) and ( analogvalue <= 2.5440 ) then
       convertspeed := 1.3913
    else if ( analogvalue > 2.5440 ) and ( analogvalue <= 2.6200 ) then
       convertspeed := 1.8233
    else if ( analogvalue > 2.6200 ) then
       convertspeed := 1.8233
end;                              { --- function convertspeed   ------------ }

function convertpitchrate ( analogvalue : real ) : real;

{ This function converts a pitchrate analog volts value to an AUV status
  parameter.  This function is derived from experimental observation.     }

const
     convertconstant = 0.125;    { conversion constant                      }

begin                             { --- function convertpitchrate   -------- }

   convertpitchrate := analogvalue / convertconstant;

end;                              { --- function convertpitchrate --------- }

procedure ClosedLoopControlScreen;
{ This procedure displays the Closed Loop Control Screen. It is displayed
  throughtout the AUV pilotting run.  It is updated with status and control
  parameters by control routines as they execute in the program sequence.   }

begin         { --------------- ActiveControlScreen ------------------ }
      clrbox2 (x1,y1,x2,y2);
      boxprint(y1+1,x1,x2,'A U V   S T A T U S  /  C O N T R O L  S C R E E N');
      boxprint(y1+2,x1,x2,'=============================================');
      boxprint(y1+4,x1,x2,'STATUS OF A U V OPERATING PARAMETERS :');

      writeln (tabxy (x1+5,y1+6),'AUV DEPTH     [ in ]        : ');

      writeln (tabxy (x1+5,y1+7),'AUV SPEED     [ ft/sec ]    : ');

      writeln (tabxy (x1+5,y1+8),'AUV PITCHRATE [ Deg / sec ] : ');
      boxprint(y1+10,x1,x2,'A U V  CONTROL STATUS');

      write (tabxy (x1+5,y1+12),'CURRENT TARGET DEPTH             : ');

      write (tabxy (x1+5,y1+13),'CURRENT A U V OPERATING MANEUVER : ');
      write (tabxy (x1+5,y1+14),'CURRENT A U V OPERATING MODE     : ');
      boxprint(y1+17,x1,x2,'THIS WILL BE THE DISPLAY DURING  A U V CONTROL');
      boxprint(y1+18,x1,x2,'STATUS WILL BE UPDATED EVERY SECOND. ');
      boxprint(y1+19,x1,x2,'NEXT :   ENTER AUV OPERATING DEPTH ');
      boxpause(x1+15,y1+21);
end;         { --------------- ActiveControlScreen ------------------ }
```

```
procedure GetTargetDepth (var tgtdepth : integer ;
                          var tgtdepthvolts : real );

{ This procedure solicits the target AUV operating depth and converts it to
  an AUV equivalent tgtdepth analog voltage and passes both of these
  parameters. }


begin        { -------- GetTargetDepth ---------------------------- }
    clrbox2 (x1,y1,x2,y2);
    boxprint(y1+10,x1,x2,'ENTER THE A U V TARGET OPERATING DEPTH');
    boxprint(y1+11,x1,x2,'>> NOTE : THE DEPTH SHOULD BE IN WHOLE INCHES <<');
    repeat
       begin
          boxprint (y1+13,x1,x2,'ENTER THE TARGET OPERATING DEPTH ');
          gotoxy (x1+33,y1+15);
          read ( tgtdepth );
       end;
    until ( tgtdepth in  [0..45]);
{ This next statement converts the integer user input target depth to an
  analog control voltage base on tank calibration test data acquired.      }

    tgtdepthvolts := 1.627 + 0.257 *  tgtdepth ;

end;         { -------- GetTargetDepth ---------------------------- }

procedure RunModeScreen;

{ This procedure displays the Closed Loop Control Screen in the RUN MODE.   }

begin        { --------------- RunModeScreen ------------------- }
      clrbox2 (x1,y1,x2,y2);
      boxprint(y1+1,x1,x2,'A U V  S T A T U S / C O N T R O L  S C R E E N');
      boxprint(y1+2,x1,x2,'==============================================');
      boxprint(y1+4,x1,x2,'STATUS OF A U V OPERATING PARAMETERS :');

      write (tabxy (x1+5,y1+6),'AUV DEPTH      [ in ]        : ');

      write (tabxy (x1+5,y1+7),'AUV SPEED      [ Ft / sec ] : ');

      write (tabxy (x1+5,y1+8),'AUV PITCHRATE [ Deg / sec ]: ');
      boxprint(y1+10,x1,x2,'A U V  CONTROL STATUS');

      write (tabxy (x1+5,y1+12),'CURRENT TARGET DEPTH : ');
      write (tabxy (x1+5,y1+13),'CURRENT MODE         : ');
      write (tabxy (x1+5,y1+14),'CURRENT MANEUVER     : ');

      boxprint(y1+18,x1,x2,
          'PRESS KEY   F1 .. TO ENTER NEW TARGET DEPTH.        ');
      boxprint(y1+19,x1,x2,
          'PRESS KEY   F2 .. TO STOP ACTIVE CONTROL AND RESET.');

      boxprint(y1+20,x1,x2,
          'PRESS  < ESC > .. TO EXIT ACTIVE CONTROL.           ');
      end;        { ---------------- RunModeScreen------------------- }
```

```
procedure UpdateRunModeScreen (updatedepth,updatespeed,updatepitchrate : real;
                                updatetargetdepth : integer;
                                updatemode: activeCONTROLmode ;
                                updateattitude : auvattitude);
{  This module updates the Closed Loop Control Run Mode Screen with updated
   display parameters. Updates occur in intervals specified by updateincrement
   interval declared in ClosedLoopControl procedure.                        }

begin       {  --------  UpdateRunModeScreen  ---------------------------  }

{    UPDATES   STATUS OF A U V OPERATING PARAMETERS               }
   writeln (tabxy (x1+37,y1+6),updatedepth:6:2);
   writeln (tabxy (x1+37,y1+7),updatespeed:6:2);
   writeln (tabxy (x1+37,y1+8),updatepitchrate:6:2);

{ UPDATES THE   A U V  CONTROL STATUS                             }
   write (tabxy (x1+30,y1+12),updatetargetdepth:2);
   case updatemode of
      run  : writeln (tabxy (x1+30,y1+13),'RUN  ');
      reset: writeln (tabxy (x1+30,y1+13),'RESET');
      exit : writeln (tabxy (x1+30,y1+13),'EXIT ');
   end;
   case updateattitude of
      maintain : writeln (tabxy (x1+30,y1+14),'MAINTAINING DEPTH        ');
      climb    : writeln (tabxy (x1+30,y1+14),'CLIMBING TO TARGET DEPTH');
      diving   : writeln (tabxy (x1+30,y1+14),'DIVING TO TARGET DEPTH  ');
   end;
end;        {  --------  UpdateRunModeScreen  ---------------------------  }


procedure GetDigitalSensoryData ( var depthanalogvolts,speedanalogvolts,
                                   pitchrateanalogvolts :real) ;

{  This procedure uses PCLAB routines to sample selected input telemetry
   channels from the AUV and digitizes these inputs and multiplies them
   by the specified gains.

   DT 2801-A / DT 707 Board set up:   channel 1  - AUV depth input
                                      channel 2  - AUV speed input
                                      channel 3  - AUV pitchrate input     }

const

{ These are artificial gains used to simulate AUV telemetry during program
  development.  One signal from a signal generator (+/- 1.25, 9 Hz,
  characteristic of the pitchrate signal) is input to all 3 input channels.
  Gains are applied to simulate the actual values.  These and their appli-
  cation in the procedure body should be removed after program development
  is completed.                                                          }

     depthgain      = 1.0;
     speedgain      = 2.0;
     pitchrategain  = 1.0;
  Convert the digitized Analog Data Values for speed, depth, pitchrate to
  analog voltage values.  The algorithm for this conversion is found in
  Appendix D of the PCLAB documentation.                                  }

   depthanalogvolts := ( depthadv * (depthpfs-depthmfs)/noc )
                                                                 + depthmfs;
   speedanalogvolts := ( speedadv * (spdpfs-spdmfs)/noc )
                                                                 + spdmfs;

   pitchrateanalogvolts := ( pitchrateadv *
                                   (pitchratepfs - pitchratemfs)/noc )
                                                                 + pitchratemfs;


end;               { ------------ procedure GetDigitalSensoryData  ------- }

procedure Errorvolts ( tdepthvolts, adepthvolts : real;
                       var derrorvolts: real;
                       var attitude : auvattitude   );

{ This module represents the "AUV Model Reference State Space."  Actual
  depth telemetry and the target depth are compared and a voltage difference
  is computed.  This difference is then "dropped" through a voltage filter
  to determine if the difference if within an acceptable tolerance, or if a
  corrective diveplane command is necessary.  A "model gain" is applied to
  the voltage difference and an errorvoltage is calculated and passed to the
  main program for dive command generation.  Although these parameters are
  single valued, in a multi-state control program these parameters could be
  implemented as arrays and the model gain array could be the result of a
  real-time program running synchronously with the main control program.   }

{ COMPUTATIONAL SIGN CONVENTION: The voltage difference is computed as the
  difference between TARGET DEPTH, or desired AUV depth, and the ACTUAL
  DEPTH.  A PLUS voltage DIFFERENCE generates down dive plane command;
  A  MINUS voltage DIFFERENCE generates an UP dive plane command.         }


const

   depthcontroltolerence =  0.1;
   modelgain             =  1.0;      { This simulates a model referenced }
                                       {  gain parameter                   }
var

    voltsdifference       : real;


begin       { ------ Errorvolts ----------------------------------------------

   voltsdifference := tdepthvolts - adepthvolts;
```

```
{ These are AUV to DT 2801-A / DT 707 hook up board channel configurations.
  conversion and computational arguments.                              }

     depthchannel   = 1;       { AUV output to DT-707 input channel assignment}
     depthpfs       = +10.0;   { Peak depth signal value                      }
     depthmfs       = -10.0;   { Minimum depth signal value                   }

     speedchannel   = 2;       { AUV output to DT-707 input channel assignment}
     spdpfs         = +10.0;    { Peak speed signal value                      }
     spdmfs         = -10.0;    { Minimum speed signal value                   }

     pitchratechannel = 3;    { AUV output to DT-707 input channel assignment}
     pitchratepfs   = +10.0;  { Peak pitchrate signal value                  }
     pitchratemfs   = -10.0;  { Minimum pitchrate signal value               }

     noc            = 4096;    { Number of Codes; conversion resolution.
                                  The DT 2801-A performs a 12 bit conversion.
                                  NOC = (2 ^ conversion bits), ie 4096          }

{ SetUpAdc and ADConTrigger PCL function arguments
                                                : p 6-8 PCL  documentation  }

     boardnum       =1;

     numa2dchan     = 3;
     timingsource   = 2;      { -- Sets a external trigger,internal clock }
     adcgain        = 1;      { Sets the A/D gain; 1,2,4,8 are options   }
     startchannel   = 1;
     endchannel     = 3;

var
   speedadv,
   depthadv,
   pitchrateadv,
   signaladv,                                 { Signal analog data value     }
   counter,status,
   chanum,i,j                : integer;

begin              { ------------ procedure GetDigitalSensoryData --------- }

{ Set up the DT 2801-A board to take data.                                  }

   status := SelectBoard (boardnum);

{ Set up the DT 2801-A board to take data from 3 input channels; Data
  sampling is initiated by the ADConTrigger single channel sample of the
  depth channel and then single ADC value samples of the speed and pitchrate
  follow.  The Trigger is connected to the DT 707 board at terminal 49 from
  a signal generating source.                                               }

            status := ADConTrigger ( depthchannel, adcgain, depthadv );
            status := ADCValue ( speedchannel, adcgain, speedadv );
            status := ADCValue ( pitchratechannel, adcgain, pitchrateadv );
```

```
{ ****************** Control voltage filter   ************************* }

{ These conditions check if depth is within tolerence.  If so a zero error
  is assigned so as to result in a zero diveplane command.                }
   if  ( voltsdifference > 0) and
             ( abs(voltsdifference) <= depthcontroltolerence ) then
      begin
         derrorvolts := 0.0;
         attitude := maintain;
      end
   else if  ( voltsdifference < 0) and
             ( abs(voltsdifference) <= depthcontroltolerence ) then
      begin
         derrorvolts := 0.0;
         attitude := maintain;
      end

{ This condition checks if actual depth is less than target + tolerence.
  In this case a DIVE  command is necessary to correct depth.             }

   else if  ( voltsdifference > 0) and
             ( abs(voltsdifference) > depthcontroltolerence ) then
      begin
         derrorvolts := voltsdifference * modelgain;
         attitude := diving;
      end

{ This last condition checks to see if the actual depth is more than target +
  tolerence.  In this case climb command is necessary to correct depth.    }

    else if  ( voltsdifference < 0) and
             ( abs(voltsdifference) > depthcontroltolerence ) then

      begin
         derrorvolts := voltsdifference * modelgain;
         attitude := climb;
      end;

end;         { ------ Errorvolts ------------------------------------------


procedure GenerateDiveplaneCommand ( divecommandvolts : real );

{ This procedure CONVERTS the analog ERRORVOLTS signal to a digital equi-
  valent voltage and sends this as a COMMAND to the AUV interface device
  for transmission as a DIVE PLANE COMMAND. It uses a single DACValue
  routine call.                                                           }

const

{ DT 2801-A DIGITAL TO ANALOG Conversion declarations                     }

   d2achannel = 0;
   pfs = 10;
   mfs = - 10;
   noc = 4096;
var
   digitaldatavalue, status          : integer;
```

```
function ConvertAnalog2Digital ( analogvalue : real ): integer;

{ This function converts analog signal volts to an equivalent
  digital value.  See App D of PCLAB book.                       }

var
   temp                : real;
begin

   temp :=  ( analogvalue - mfs ) * ( (noc - 1) / (pfs - mfs ) );
   convertanalog2digital := round ( temp );

end;

begin  { ------------- GenerateDivePlaneCommand ------------------- }

   digitaldatavalue := convertanalog2digital ( divecommandvolts );
{   status := initialize;  }
   status := selectboard (1);
   status := dacvalue ( d2achannel, digitaldatavalue );
{   status := terminate;    }

end;  { ------------- GenerateDivePlaneCommand ------------------- }


procedure InitializeParameters ;

{ This procedure initializes all declared control and display parameters to
  zero.                                                                        }

begin                             { ---- procedure InitializeParameters ---    }

     auvdepthvolts := 0.0;
     auvspeedvolts := 0.0;
     auvpitchratevolts := 0.0;
     auvdepth := 0.0;
     auvspeed := 0.0;
     auvpitchrate := 0.0;
end;                              { ---- procedure InitializeParameters ---    }
```

```
begin               { ----------------- ClosedLoopControl   ------------- }

activemode := run;
initializeparameters;

repeat           { ------------- Repeat until activemode = exit   ------ }


   repeat           { ---- Repeat until activemode = reset        --------- }

      ClosedLoopControlScreen;
      GetTargetDepth ( targetdepth, targetdepthvolts );
      RunModeScreen;

      while ( not  keyhit ( modereply, modereply2))   do
{NOTE:   THIS IS THE PROGRAMMABLE INTERRUPT WHICH IS MASKED BY THE PCLAB
 ROUTINES. THE USER MUST USE A  CONTROL-BREAK  < CNTRL^> TO STOP PROGRAM
 EXECUTION AND EXIT THE PROGRAM.                                        }


         begin
            updatecounter := 0;

            while ( updatecounter <  updateincrement ) do
              begin
               GetDigitalSensoryData (auvdepthvolts,auvspeedvolts,
                                            auvpitchratevolts );

               Errorvolts ( targetdepthvolts, auvdepthvolts ,
                            deptherrorvolts, attitude);


               GenerateDiveplaneCommand ( deptherrorvolts );

               updatecounter := updatecounter + 1;


             end;   { while updatecounter < updateincrement   }

            auvspeed := convertspeed ( auvspeedvolts );
            auvdepth := convertdepth ( auvdepthvolts );
            auvpitchrate := convertpitchrate ( auvpitchratevolts );

            UpdateRunModeScreen ( auvdepth, auvspeed, auvpitchrate,
                                  targetdepth, activemode, attitude );

      end;          { while not KeyHit                       }

      if (ord(modereply) = 27) and (ord(modereply2) = 59) then
        activemode := run                                  { KeyHit=F1 }
      else if (ord(modereply) = 27) and (ord(modereply2) = 60) then
        activemode := reset                                { KeyHit=F1 }
      else if (ord(modereply) = 27) and (ord(modereply2) = 0) then
        activemode := exit;                               { KeyHit= ESC}

   until ( activemode = reset ) or (activemode = exit );

until ( activemode = exit );
```

```
end;          { -------------------- ClosedLoopControl  ---------------- }

begin         { ---------------------- StatusAndCommand ----------------- 

   repeat
      StatusAndCommandScreen;
      GetKey ('',mode,mode2);
      if ( ord (mode) = 27 ) and ( ord (mode2) = 59 ) then
         begin
              clrbox2 (x1,y1,x2,y2);
              ClosedLoopControl;
         end;
   until ( mode in ['Q','q'] );

end;          { ---------------------- StatusAndCommand -----------------

procedure InitializeZeroDigitalSignalOut;

{ This procedure MUST be executed as the first procedure called in the main
program to insure a zero signal out on the 2 output channels.  Otherwise the
DT 2801-A board defaults to a minimum full scale output.                 }

const
   digitalchan0         = 0;
   digitalchan1         = 1;
   digitalcommandboard  = 1;

var
   status,
   digitaldatavalue     : integer;

begin
   digitaldatavalue := 2048;      { This will be converted to an equivalent
                                        zero analog signal out on a 12 bit
                                        resolution converter like DT 2801-A.   }

   status := initialize;
   status := selectboard ( digitalcommandboard );
   status := dacvalue ( digitalchan0, digitaldatavalue );
   status := dacvalue ( digitalchan1, digitaldatavalue );
   status := terminate;
end;

procedure DeactivateADBoardAndExitProgram;

{ This procedure deactivates the DT 2801-A board and presents
  an exit screen.                                                    }

var
status              : integer;

begin  { ---------------- DeactivateADBoardAndExitProgram ---------------- }
  status := terminate;
  clrbox2 (x1,y1,x2,y2);
  boxprint (y1+10,x1,x2,
                 'THIS CONCLUDES YOUR AUV AUTOPILOTTING SESSION , BYE');

end;  { ------------------- DeactivateADBoardAndExitProgram ---------------- }
```

```
BEGIN         { ----------------------------      MAIN PROGRAM    ---------- }

   InitializeZeroDigitalSignalOut;
   clrscr;
   repeat

      MainMenu ( option );
      if ( option in ['R','r']) then
          begin
             repeat
               begin
                   StatusAndCommand ( controlmode );
               end;
             until ( controlmode in ['q','Q']);
          end;
   until ( option in ['Q','q']);

   DeactivateADBoardAndExitProgram;

END.          { ------------------------------- MAIN PROGRAM    ---------- }
```

LIST OF REFERENCES

1. 5th International Symposium on Unmanned Untethered Submersible Technology, University of New Hampshire, June 22-24, 1987, Symposium Proceedings.

2. Boncal, R.J., A Study of Model Based Maneuvering Controls for Autonomous Underwater Vehicles, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1987.

3. Brunner, G.M., Experimental Verification of AUV Performance, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1988.

4. User Manual for DT 2801 Series, Single board, Analog and Digital I/O Systems for the IBM Personal Computer, Data Translation, Inc., Marlborough, Massachusetts, 1986.

5. User Manual for PCLAB, SP041 v 2.00, Data Translation, Inc., Marlborough, Massachusetts 01752-1192.

6. DT/NOTEBOOK, Laboratory Technologies Corporation, Wilmington, Massachusetts 01887, 1986.

7. MATRIXx, Integrated Systems Inc., Santa Clara, California, 1987.

8. Turbo Pascal Reference Manual, and Turbo Pascal Compiler v3.0, Borland International Inc., Scotts Valley, California, 1985.

9. Rugg, Tom and Feldman, Phil, Turbo Pascal Program Library, Que Corporation, Indianapolis, Indiana, 1986.

10. Borrie, John A., Modern Control Systems: A Manual of Design Methods, Prentice-Hall International, Englewood Cliffs, New Jersey, 1986.

11. Friedland, Bernard, Control Systems Design: An Introduction to State-Space Methods, McGraw-Hill, New York, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code 69Hy Mechanical Engineering Department Naval Postgraduate School Monterey, California 93943-5000	5
4.	Professor D.L. Smith, Code 69Sm Mechanical Engineering Department Naval Postgraduate School Monterey, California 93943-5000	1
5.	Professor R. McGhee, Code 52Mz Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	1
6.	Professor R. Christi, Code 62Cx Electrical and Computer Engineering Department Naval Postgraduate School Monterey, California 93943-5000	1
7.	Dr. G. Dobeck (Code 4210) Head, Navigation and Guidance NCSC Panama City, Florida 32407-5000	1
8.	Russ Werneth, Code u25 Naval Surface Weapons Center White Oak, Maryland 20910	1
9.	Paul Heckman, Code 943 Head, Undersea AI & Robotics Branch Naval Ocean System Center San Diego, California 92152	1

10.	Dr. D. Milne, Code 1563 DTNSRDC, Carderock, Bethesda, Maryland 20084-5000	1
11.	RADM G. Curtis, USW PMS-350 Naval Sea Systems Command Washington, D.C. 20362-5101	1
12.	LT Relle L. Lyman, Jr., USN Code 90G Naval Sea Systems Command Washington, D,C. 20362-5101	1
13.	LT Richard Boncal, USN Raynes Neck Road, RFD 2 York, Maine 03909	1
14.	Distinguished Professor G. Thaler, Code 62Tr Electrical and Computer Engineering Department Naval Postgraduate School Monterey, California 93943-5004	1

FILM

8